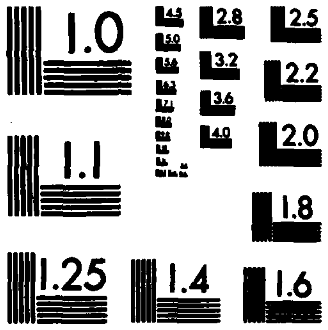
AD-A154 466   AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING   1/2
AIR LAUNCHED CRUI..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..   R J MILLAR
UNCLASSIFIED   DEC 84 AFIT/GCS/ENG/84D-17   F/G 9/2   NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

DTIC

1

AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING AIR LAUNCHED CRUISE MISSILE MISSIONS

Thesis

AFIT/GCS/ENG/84D-17

Robert J. Millar
Capt USAF

Ac
NT
DT
Un
Ju
By

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/84D-17

(1)

AN
ARTIFICIAL INTELLIGENCE BASED FRAMEWORK
FOR PLANNING
AIR LAUNCHED CRUISE MISSILE MISSIONS

Thesis
AFIT/GCS/ENG/84D-17

Robert J. Millar
Capt USAF

Approved for public release; distribution unlimited

AFIT/GCS/ENG/84D-17

AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING AIR LAUNCHED CRUISE MISSILE MISSIONS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Robert J. Millar, B.S.
Capt USAF
Graduate Computer Science

December 1984

PREFACE

This thesis was undertaken for two reasons. Previously, I had been involved in some research in the area of ALCM mission planning and was almost overwhelmed by the complexity of the problem. Any type of planning with a computer is difficult, but, ALCM mission planning provides an extremely difficult problem domain. Yet, I personally believe a system which plans ALCM missions is needed within the Department of Defense and I am convinced one will be built. Thus, one purpose of this thesis is to hopefully influence the architecture of any future ALCM mission planning system developed.

The second purpose of this thesis was to provide a test bed for experimenting with several ideas and techniques currently being promoted by the artificial intelligence community. These ideas and techniques seem very powerful, yet I had doubts as to how practical they were. By the time the thesis had been finished, I had concluded that the method of implementation was the deciding factor in how practical these ideas were.

I wish to thank Capt Rob Milne, my thesis advisor, and Capt Steve Cross, my thesis reader, for their assistance. They have spent many hours listening to my ideas and have helped me significantly to organize and clarify my thoughts on this thesis.

Robert J. Millar

Table of Contents

List of Figures

AFIT/GCS/ENG/84D-17

Abstract

This thesis proposes a design for a computer system capable of generating mission plans for an Air Launched Cruise Missile (ALCM)~~. This proposed design is~~ based upon recent research in the field of Artificial Intelligence and has three major features of significance.

A hybrid blackboard architecture, called a dual blackboard, is developed ~~in this thesis~~ as the basic architecture of ~~this~~ the planning system. ~~This~~ The hybrid architecture emphasizes the parallel processing potential inherent in a blackboard system, and ~~In addition, it~~ forces some structure onto the blackboard concept, preventing a blackboard from being implemented as simply a global variable or common block.

The knowledge source performing the actual mission planning is designed as a rule based production system, which ~~This~~ allows ~~the~~ mission planners to build the rules used for the planning process. These rules define for the system what makes a mission plan undesirable and what can be done to make the ~~that~~ mission plan more acceptable. As the mission planners specify the rules used, this system should be viewed as ~~more of~~ a ALCM planning tool, akin to a spreadsheet program, instead of a blackbox system used for planning ALCM missions.

cont

A modified meta-planning approach ~~to planning~~ is used ~~in~~ conjunction with several other techniques to reduce the number of alternative mission plans to ~~that must~~ be evaluated. before the best plan can be identified. Domain specific constraints are applied against proposed mission plans proposed by the system to ensure only those ~~mission~~ plans satisfying all constraints are even considered. ~~for evaluation.~~ and Heuristics are used to rank those ~~mission~~ plan alternatives satisfying all the constraints, ensuring only the most promising get evaluated. The heuristic used in this design favors those mission plans that best approximate a nominal mission plan - one with a direct flight path.

AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING AIR LAUNCHED CRUISE MISSILE MISSIONS

I. Introduction

The field of Artificial Intelligence (AI) has recently begun to receive considerable attention due to the success of a new class of computer program called an expert system. These expert systems have seen successful use in extremely difficult applications - where conventional approaches for computer solutions often fail. The planning of Air Launched Cruise Missile (ALCM) missions is such an application. The ideas and techniques developed in the AI labs in the past decade provide a framework within which a computer system for planning ALCM missions can be successfully developed.

Background

In December 1982, the United States Air Force placed the first ALCM's into operational status (12:46). The ALCM is a small, pilotless, self-guided, air-breathing vehicle, with short wings, that flies like an airplane at sub-sonic speeds. An onboard computer provides the guidance control necessary for the vehicle. The ALCM is launched from a flying platform, currently B-52 bombers, and delivers a nuclear warhead against targets up to about 1500 n.m. from the launch point (12). Because of it's nuclear warhead, the

ALCM is classified as a strategic weapon. Thus, the people who currently plan the missions for this country's strategic bomber force must now take on the additional responsibility of planning the ALCM missions.

The deployment of the ALCM marks the beginning of what will be an explosion in the workload of this country's strategic planners. There are several reasons why the workload will increase - including the increased number of missions to be planned. In the past, these planners prepared plans for a small strategic bomber force which remained fairly constant in size from year to year. It is estimated 328 bombers were assigned strategic missions in 1983 (14:694). With the deployment of the ALCM, the planners must develop an increasingly large number of missions every year. The Air Force currently plans to buy 1739 ALCM's for deployment in the 1980's (12:47). This represents over a 600% increase in the number of strategic missions to be planned by 1990.

Another factor which will increase the workload for the planners is the added complexity involved in planning ALCM missions. Since the ALCM is a pilotless vehicle, mission plans must be specified in greater detail. For example, consider when a planner wishes a vehicle to fly at 100 ft above ground level (AGL). For a bomber mission, the planner can simply state the requirement. The bomber pilot will ensure the climbs and descents are started when necessary to maintain the 100 ft AGL clearance. However, for an ALCM mission, the mission plan must include when to start and stop all climbs and descents necessary to maintain the 100 ft clearance. This requires the mission planner consider more carefully both vehicle performance and terrain roughness when preparing mission plans.

A third factor which will increase the mission planner workload is anticipated system improvements. Once the original 1739 ALCMs are deployed, a new generation of ALCMs will begin deployment in the 1990's (11). The number to be built and the precise capabilities of this new generation ALCM has not yet been established. But, it is certain to result in increased workload for the mission planners. Thus, for the foreseeable future, the workload of the strategic planners will be constantly increasing.

Unfortunately, these planners are not prepared today to handle this projected increase in workload. The mission planners have massive amounts of computer support available for evaluating the quality of the missions planned. But, there exists minimal computer support for the actual planning of the missions. This remains primarily a human endeavor. Continuing this approach to mission planning by increasing the number of planners to handle this increased workload would be expensive and subject to the laws of diminishing return. What the planners need is a computer system which suggests suitable ALCM mission plans to the planner. Then the planner can evaluate the suggested mission plan and use those determined to be acceptable.

Actually, this increased workload may very well force a positive change in the the way strategic planners perform their missions. With a computer system available to assist with the mission planning, the mission planners can begin devoting more of their time to evaluating mission plans and devising new tactics. If this occurs, then both the number of missions which can be planned as well as the quality of those missions will increase.

Problem Definitation

The problem of planning an ALCM mission consists of finding an optimal flight path from a specified launch point to a designated target which minimizes both the probability of destruction due to enemy defenses and the probability of crashing. At the same time, the flight path must maximize the probability of hitting the target. These factors are all interdependent, in that altering one will often cause the others to be altered (8:158). Additionally, the selected flight path must satisfy a large number of strategic constraints. This usually results in a sub-optimal flight path being selected. The completed mission plan consists of a list of action points; where each action point identifies a change in the vehicle's heading or altitude necessary for the vehicle to fly the selected flight path. Chapter two describes in detail the factors involved in planning an ALCM mission.

Unfortunately, planning is one of those domains which has proven very difficult to do with a computer. Because of this, considerable research on planning has been done within the AI community (3). This research is reviewed in Chapter four and provides a framework around which a successful ALCM mission planning system can be developed.

Thesis Objective

The objective of this thesis was to design a prototype computer system capable of suggesting suitable ALCM mission plans to a mission planner. The difficulty with planning systems is the large number of alternatives which must be explored before a choice can be made. The number of alternatives to be explored can be reduced significantly by using current AI technology and exploiting current AI research on planning. Therefore, the primary objective was to develop an AI based framework within which ALCM missions can be planned. Additionally, prototype software was developed to demonstrate the feasibility of the designed framework. The end product of this thesis is the recommendations on software architecture for future production versions of ALCM mission planning software. These recommendations are based on the successes and failures of the prototype software.

Scope

Because of the experimental nature of this thesis, neither the proposed design or the prototype software developed is suitable for production use. The primary objective was to be able to evaluate the effectiveness of the proposed AI based framework for mission planning - not to generate ALCM mission plans. The time available for this thesis simply did not allow a complete ALCM mission planning system to be developed. Therefore, the scope of the problem was significantly reduced in several manners.

To assess the quality of an ALCM mission requires several different computer models required for both mission planning and the evaluation of the mission plans. Development of even one of these models, in a comprehensive manner, was not feasible in the available time. Therefore, simplified models were developed for use in this thesis.

A production mission planning system must have a host of features available to ensure a good user-interface between the system and the mission planner. These features would include at least a natural language interface and some type of explaination/justification features for explaining to the user why the system selected the flight path proposed to the mission planner. However, for this thesis, the user interface consists of a limited set of keywords and menus.

Due to limited availability of digitized terrain data, the modeling necessary to determine the probability of crashing into the ground is not done in this thesis. While this would be unacceptable in a production system, it does not invalidate the approach used in this thesis. The amount of computation required to evaluate each alternative plan is of less importance in the design of a planning system than the number of alternatives which must be considered. This thesis concentrates on reducing the number of alternatives which must be considered.

An ALCM mission plan is subject to multiple strategic constraints. However, only the "Time of Arrival" constraint is addressed in this thesis. This constraint is described in detail in chapter two. Other constraints could be implemented in a similiar manner in a production ALCM mission planning system.

Assumptions

The major assumption made for this thesis involves the current mission planners. It is assumed there is a logical and identifiable reason for planners preferring one route over another. Because of the large number of reviews which occur before a mission is finalized, the author believes this to be true. These reviews result in the subjective factors in a mission plan largely being removed. Precisely because the review process removes much of the subjective nature of the planned mission, the author believes computers can perform the mission planning task.

The other assumption made for this thesis involves the limited scope of this thesis. It is assumed the limited scope does not invalidate the approach used in this thesis. The precision of calculations should not be of concern for a prototype system like this. More important is how the results of those calculations are used to solve the desired problem. If time permitted, a complete system suitable for planning ALCM missions could have been developed. But that would not have altered any conclusions reached on system architecture; only the quality of the actual mission plans.

Summary of Current Knowledge

During the early 1980's, two R&D systems were built for ALCM mission planning. One system was developed by Boeing Aerospace Company and used current AI technology. The other system, developed by Systems Control Technology used more conventional Operations Research techniques.

Boeing Aerospace Company. Boeing Aerospace Company, of Seattle Wa, has developed prototype ALCM mission planning software based on the AI technology, expert systems. This was an internally funded research project built around an OPS-5 production system (9:9). A measure of merit was given given to each route segment as it was generated. This measure of merit was assigned by assessing penalties to those route segments which varied significantly from a nominal route. Those route segments with lowest measure of merit were expanded further. This tends to limit the number of possible routes evaluated before a satisfactory route is selected (9:11). Unfortunately, fes details have been published on the architecture of the system. And the only comment on system performance was the statement "Expert systems based on production rules have been found to be capable of satisfying these requirements" (9:9).

Systems Control Technology. The only other known attempt to automate the planning of ALCM missions was performed by Systems Control Technology (SCT), of Palo Alto Ca., as a R&D effort under Defense Advanced Research Projects Agency sponsorship. This effort included more than just the planning of ALCM missions - but only that aspect of their effort is of interest for this thesis.

The approach used by SCT was to divide the area between the ALCM launch point and target into equal sized squares - usually with 5 n.m. edges. For each square, eight route segments were built (every 45 degrees) from the center of the square to the edges of the square. Assigned to each of the eight route segments of the square was a cost for a vehicle to traverse that route segment. This cost was based on terrain roughness and the threat density of the area being traversed (10). Figure No. 1 shows six such squares with the traversal costs for all route segments of each square. The least cost path from point A to point B in Figure No. 1 has a total cost of 18.

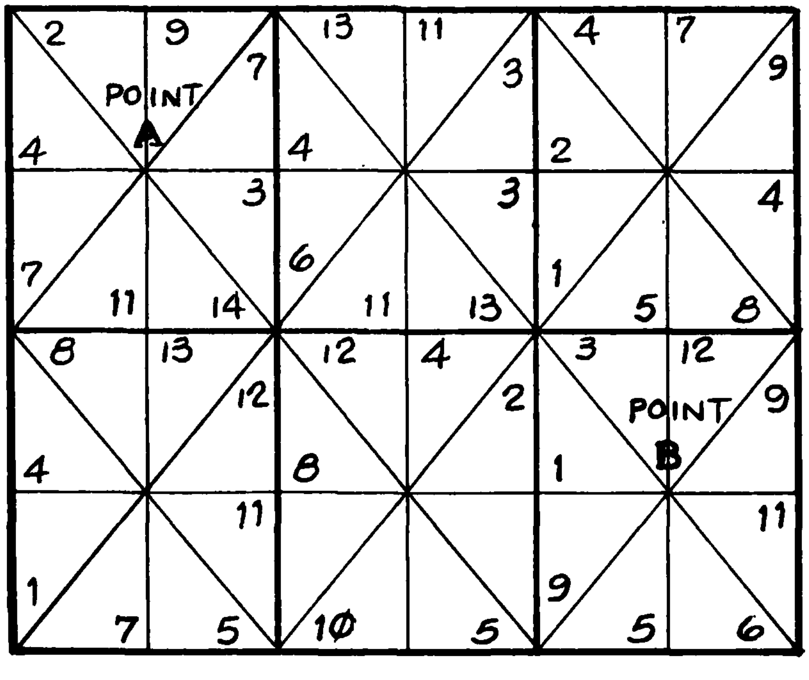


Figure No. 1 - ALCM Route Segments With Assessed Costs

Once all route segments are built and traversal costs assessed, the least cost path between all launch points, targets, and navigation update points are determined. Then the least cost path from launch point to target is determined by selecting those navigation update points which provide the least cost path from launch point to target (10).

There are two primary shortcomings with the approach used by SCT. First, the computational costs for realistic scenarios may be prohibitively high. The current ALCM has a range of about 1500 n.m. (12:46). Assuming only a 100 n.m. corridor for maneuvering and a 5 n.m. grid size, each ALCM route must consider 6,000 grids. These 6,000 grids represent 48,000 route segments to be evaluated.

More bothersome than the number of route segments to be evaluated, is the number of route segments that must be considered before selecting a route. Disallowing backtracking, each grid can be departed in one of five directions. Thus, for the example in the previous paragraph, 5^{6000} possible routes must be considered to find the best route; assuming no intermediate navigation update points.

Of course intermediate navigation update points reduce the size of the problem considerably. In fact, SCT depends upon the constraining effect of navigation update point requirements to reduce computational costs. In the previous example, if there were two navigation update points, equally spaced, then only $(3*5)^{2000}$ routes would require consideration before the best route could be selected. While this is a significant reduction in the alternative routes to be considered, it is still too many.

The other area of concern centers around the dynamics of mission planning. The SCT approach is static in nature. The costs for all route segments are determined at the start of all ALCM missions and never updated. Yet, in reality, the costs of various route segments varies with time. For a given route segment that is protected by an enemy defense site, the traversal cost may be moderately high before any ALCM sorties are flown. However, at some point that defense site may be destroyed. When this occurs the traversal cost of that route segment becomes more acceptable. Consider also a route segment which goes past the site of a nuclear detonation. For a short period after that detonation the traversal cost of that route segment will be prohibitively high - regardless of how attractive the route segment may have looked at the beginning of the scenario.

While the SCT effort has its limitations, the results are impressive. Any effort in the area of automated ALCM mission planning should study the SCT effort carefully. It have provided the starting point for this thesis.

Summary

With the deployment of the ALCM, the strategic planners for this country face a tremendous increase in workload. A computer system which can suggest appropriate mission plans for specified sorties would not only allow the mission planners to meet this increased workload, but to improve the quality of the mission plans by providing the mission planners more time to evaluate mission plans and develop better tactics.

Unfortunately, planning systems have always been difficult to develop. One of the main difficulties involved is the large number of alternatives which must be considered before a choice can be made. By utilizing the latest AI research, the ALCM mission planning system designed in this thesis will provide better plans for fewer alternatives explored than previously considered possible.

II. ALCM Mission Planning

The objective of ALCM mission planning is to identify that flight path from launch point to target which maximizes the probability of the ALCM arriving on target. This means the mission planner must identify the flight path which will satisfy the unique navigation requirements of the ALCM and minimize the probability of the vehicle being destroyed en-route to the target. Additionally, the planner must satisfy several strategic constraints. These constraints may be imposed for reasons of strategic doctrine or to coordinate the mission with other strategic missions being planned. Each ALCM mission, while individually planned, is a part of a global strategic plan. As such, there can be considerable interaction with other missions.

Navigation Requirements

The navigation requirements for the ALCM are fairly rigid and have significant impact on the flight paths that can be used by an ALCM. These requirements are necessary because of the relatively slow speeds, high sub-sonic, at which the ALCM flies. These slow speeds introduce navigation error for two different reasons. First, the slow speeds forced the ALCM to use an Intertial Navigation System (INS) for onboard navigation. Unfortunately, a INS is subject to drift errors. Secondly, at the slow speeds the ALCM flies, the wind and weather can significantly effect the flight path of the vehicle and must be compensated for (16:21).

INS Drift Error. An inherent characteristic of an INS is that it is subject to drift error. This drift error results in the heading indicator slowly changing even when the vehicle has not changed heading. Therefore, when a computer is maintaining a constant vehicle heading, as in the case of a ALCM, the drift error results in the vehicle flying off course. In a commercial INS the drift error can be up to one mile per hour of flight time. Additionally, severe changes in the vehicle's flight attitude can accelerate the drift error of the INS onboard the vehicle (13:36). To minimize the drift error, the INS must be periodically updated from an external source.

Wind Drift Error. An ALCM flying at sub-sonic speeds is affected by the local air currents. Thus the direction and speed of the wind directly impact the flight path and ground speed of the vehicle. Assuming a maximum wind of twenty miles per hour, without compensation the ALCM could be anywhere within a twenty mile radius of the target after an hour of flight time. The target could easily be missed after only one hour of flight time - and ALCM flights are usually several hours in duration. Thus, the direction and speed of the wind must be identified and compensated for. This can be done by comparing the vehicle position as determined by the on-board navigation system against an external source. Any difference, once the INS has been updated, is the result of wind drift and must be compensated for.

__TERCOM Guidance Scheme__. The terrain contour matching (TERCOM) guidance scheme was developed to provide the external source necessary to correct for INS and wind drift. The TERCOM guidance system consists of a radar altimeter and computer. As the ALCM flies over terrain a radar altimeter monitors the altitude of the terrain. The current terrain altitudes are compared against maps, called TERCOM maps, stored in the onboard computer. When a specified number of terrain elevations can be matched against a TERCOM map the current location of the ALCM has been identified. Then the INS can be updated and the vehicle heading corrected to compensate for any wind (15:36-37).

Figure No. 2 demonstrates how this process works. The solid line indicates the planned flight path and the dashed line shows the flight path actually flown. When launched, the ALCM was placed on a heading intended to intersect with the middle of the TERCOM map. When the middle of the TERCOM map was reached the ALCM was to change heading to the left and proceed to the target. However, due to INS and/or wind drift the ALCM crossed the TERCOM map farther to the left than expected. By the time the ALCM reached point A, the onboard computer identified it's current position. Then the INS was reset to eliminate any INS drift error. After the INS was reset, the onboard computer calculates the direction and strength of the current winds by comparing the launch location and time against the current location and time. Once the current winds are calculated, the onboard computer calculates the heading necessary to reach the target from the ALCM's current location and compensentating for the current winds.

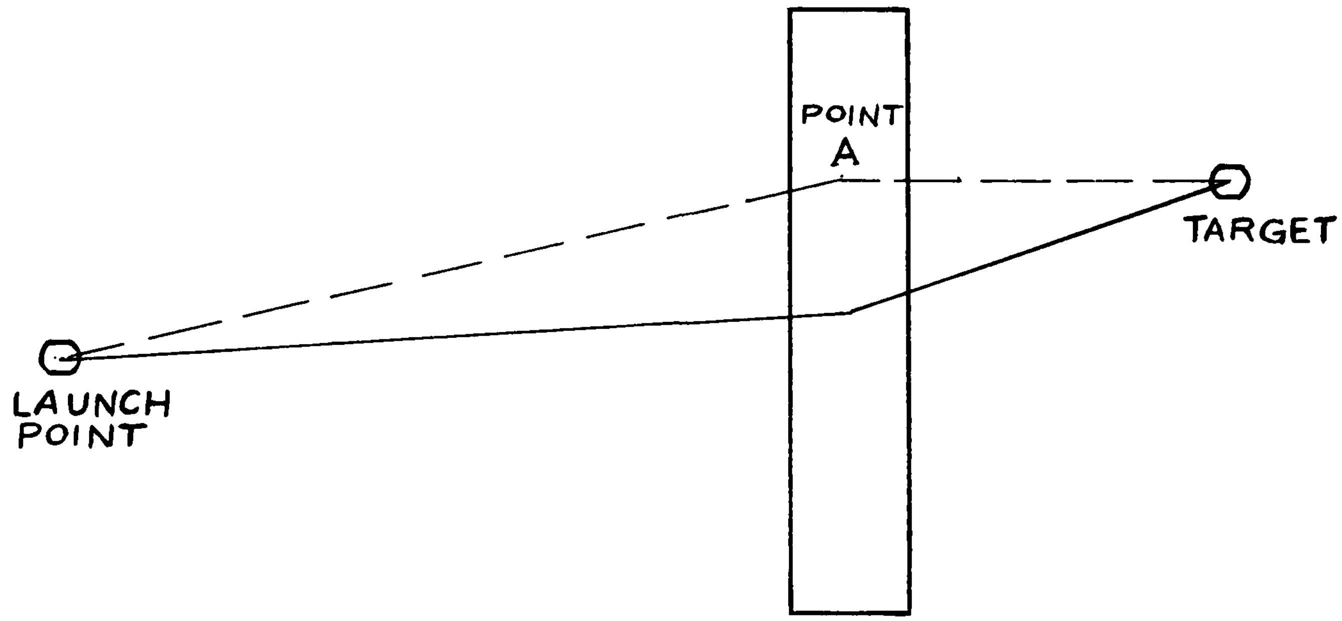


Figure No. 2 - ALCM Flight Path with TERCOM Updating

While the TERCOM guidance scheme does ensure the ALCM can find the target, it also places severe restrictions on the possible flight paths to be used. This guidance scheme requires the planned flight path of an ALCM cross the center of TERCOM map areas on a periodic basis. However, there are only a limited number of TERCOM map areas available. These TERCOM map areas are limited in number because each must be unique in altitude changes from any other land mass in that region of the world. Otherwise, the potential would exist for an ALCM to fly over one TERCOM map and identify it as another. The limited number of TERCOM maps often force an ALCM planner to select indirect routes to the target in order to satisfy navigation requirements (15:36-40).

Threats To Vehicle Safety

The primary threats to an ALCM enroute to the target are the enemy defenses and the chances of flying into the ground, for which the term "clobber" has been applied. Both threats are interrelated as flying the ALCM higher above the ground decreases the chances of flying into the ground, but increases the vulnerability of the ALCM to enemy defenses. The converse is also true. Flying the ALCM closer to the ground decreases the vulnerability to enemy defenses, but increases the chances of flying into the ground. Therefore, simultaneous minimization of both threats must be done to find a optimal flight path. However, attempts to minimize these threats must be tempered by the fact that each change in altitude reduces the range of the ALCM (8:158).

Enemy Defenses. About 7000 ground based radar sites surround the Soviet Union (4). These radar sites are operated in conjunction with either Surface-to-Air Missile (SAM) sites or ground controllers capable of vectoring a fighter aircraft to an intercept position. Thus, all radar sites are potentially lethal, with the level of lethality determined by the type of system the radar site is used with. To minimize the threat of enemy defenses, radar detection must be minimized.

Radar works on a line-of-sight principle. This means a radar site cannot see an object which is below the horizon. A vehicle can remain below the horizon by either flying low or using prominent land forms to shield the vehicle from view (for which the term terrain masking is used). Flying close to the earths surface allows the vehicle to take advantage of the curvature of the earth. As can be seen in Figure No. 3, one vehicle can be shielded from the radar site by the earth's curvature while another vehicle at the same location is not shielded because it is flying at a higher altitude.

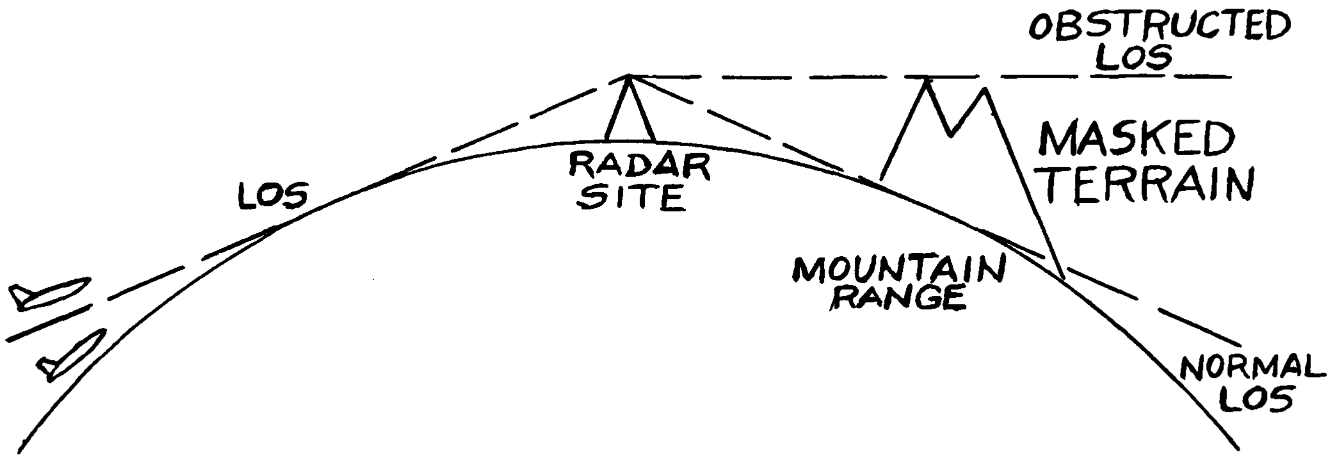


Figure No. 3

The effects of terrain masking encourage a mission planner to look for terrain features which can be used to shield the ALCM from radar sites. Figure No. 3 shows how a vehicle being shielded from detection by terrain features can fly significantly higher above the terrain elevation. This eliminates the danger of clobber. So, terrain masking provides the only method available that simultaneously reduces both major threats to the ALCM.

<u>Clobber</u>. From the previous discussion, one concludes the lowest flight path possible provides the best strategy for avoiding radar detection when terrain masking is not

available. However, the lower the flight path; the higher the probability of clobber (crashing into the ground). Thus it appears desirable to identify that minimal altitude which minimizes the probability of clobber and use that altitude for the entire flight path.

However, the ALCMs' performance does not always permit this. Often terrain will increase or decrease in altitude faster than the ALCM can change altitude. When approaching terrain which is rapidly increasing in altitude the ALCM must begin climbing early. Otherwise, the vehicle will fly into the side of the mountain or hill. Conversely, when following terrain decreasing in altitude which levels off rapidly the ALCM must level off sooner that the terrain to prevent sinking into the ground. In Figure No. 4, the solid line is a optimum constant altitude above ground level; the dashed line is the flight path that must be used to safely fly over the terrain. Notice every deviation from the optimal altitude places the vehicle higher and more exposed to enemy defenses.

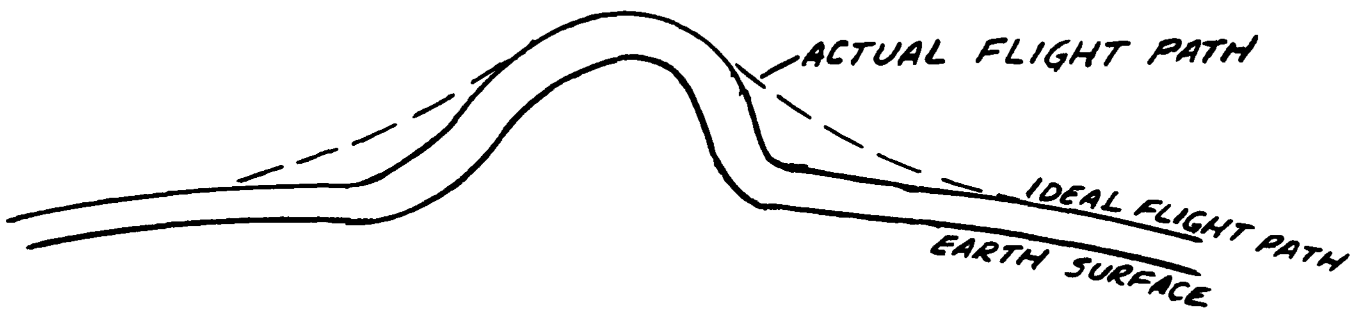


Figure No. 4 - ALCM Flight Path Minimizing Clobber

Conflict Resolution. If ALCM deployment proceeds as scheduled, strategic planners should be planning missions for over 1700 ALCMs by 1990. Combined with bomber missions, this means the mission planners will be planning missions for over 2000 vehicles. These vehicles will be flying over same part of the world at approximately the same time. Some may have common targets and it is almost certain that some of these vehicles will have crossing flight paths. Thus, the strategic planner must take steps to prevent collisions between these vehicles. Those vehicles that cross flight paths slowly or have parallel flight paths will require constant attention during the planning of the second flight path. The mission plan of the first vehicle may need to be modified to ensure proper clearance between the vehicles.

Fratricide. The term "fratricide" has been used to describe destruction of a vehicle by friendly fire. The strategic mission of the ALCM calls for the vehicle to fly thru an active nuclear environment. If the vehicle were to overfly the detonation point of another sortie at the time of detonation, or shortly thereafter, the ALCM would be destroyed. Therefore, to avoid any danger from fratricide, the mission planner must avoid any area where a nuclear detonation has occured until well after the detonation.

Strategic Constraints

In addition to the constraints which must be satisfied to ensure the ALCM reaches the target, many other strategic constraints must be satisfied. These strategic constraints are imposed due to national policy or strategic doctrine. As such they are subject to frequent change. Regardless of why a constraint is imposed upon the mission planner, each newly added constraint tends to reduce the probability of the ALCM successfully reaching it's target. As examples of the types of constraints that may be imposed on an ALCM planner, some hypothetical constraints are discussed.

Time Of Arrival. A mission plan may require that the ALCM arrive at the target, or some intermediate point along the route, at some predesignated time. This may dictate a more direct flight path than the optimum flight path. Or it may require the ALCM either fly slower or take a more indirect route. In either case, the vehicle's exposure to radar will be increased and the probability of the ALCM reaching the target will be decreased.

Keep Out Zones. Due to national policy, it may be desirable to prohibit nuclear weapons from flying over certain areas. This would require the planner to route the ALCM around the restricted area. However, the keep out zone may be so large that the ALCM cannot go around the zone and still reach it's target. In that case, a different launch position may need to be used which makes it easier to avoid the restricted area.

Limited TERCOM Map Use. For stragegic reasons, it may be desirable to limit the number of ALCM sorties which use the same TERCOM map. An error with the heavily used TERCOM map or an unexpected enemy defense at that location could cause mulitple sorties to be unexpectably destroyed before reaching their targets. This type of constraint adds a new level of difficulty to the mission planners task. This is a resource allocation problem. To allocate the TERCOM map useage to those sorties who best use it requires detailed information about all previously planned sorties that use the TERCOM map. Additionally, it requires the mission planner have the option of changing a mission plan which has been completed.

Summary

The effects of INS error and unpredictable local winds requires the ALCM receive periodic navigation updates from an external source. The TERCOM guidance system was developed to provide these external navigation updates, but, this system greatly constrains the flight paths which an ALCM can use. Therefore, planning an ALCM mission consists of finding the optimum set of TERCOM maps to use given the best paths between the various TERCOM maps.

The best path between TERCOM maps is that path which minimizes the chances of the vehicle being destroyed by either enemy defenses or flying into the ground. At the same time, other factors like conflict resolution and fratricide must be considered when selecting that best path.

III. Artificial Intelligence

Artificial Intelligence (AI) has been defined as the study of how to make computers do things for which, at the moment, people are better (19:1). Planning is one area where people currently are better than computers - especially ALCM mission planning. This chapter provides an introduction to some of the basic AI techniques used in this thesis. Those readers with a sound background in AI techniques may wish to skip this chapter.

Symbolic Processing

One of the basic ideas of AI is the use of symbolic processing. Symbols are processed much like numbers in the conventional use of computers. However, individual symbols can represent more information than individual numbers. This additional information provides additional capability. If any single characteristic identifies the framework used in this thesis as AI based, it would be the use of symbolic processing.

Expert Systems

Expert systems are AI based computer programs designed to solve problems usually falling into one of the following categories: interpretation, prediction, diagnosis, design, debugging, planning, monitoring, repair, instruction, or control (7:4). Lack of an algorithm with which to solve these problems has hampered computer solutions in the past.

However, expert system solutions are not based on the use of algorithms. Instead, they attempt to reason about a problem with symbolic processing, in a manner similiar to the way a human expert would attempt to solve the problem (6:39). Often this means exploiting information about the problem domain so as to reduce the difficulty of the problem.

Architectural Principle. Expert systems are designed around an architectural principle known as separation of the inference engine from the knowledge base. This principle mandates the program (inference engine) contain little or no information about the problem. The knowledge base contains all of the detailed information about the problem, including the rules on how to solve the problem. Also included in the knowledge base would be any "heuristics" (rules-of-thumb) a human expert would use to solve the problem. The inference engine simply applies the pertinent rules to the information available. Therefore, the primary effort involved in building an expert system consists of building up the knowledge base. As a result, the term "knowledge engineering" has been applied to the building of expert systems (5:6-9).

Production System. There are several methods of implementing the architectural principal of expert systems. One of the most popular implementations is called a production system. A production system consists of three major components. The first component, a local database, contains those facts known to be true or false at the current time.

The second component is a rules database containing production rules. These production rules are statements of the form "if condition then action" (1:190). Each rule describes how a particular process is performed. Because of their form, production rules are easily understood by those individuals who are not computer experts. Thus, it becomes easier for those individuals to transfer their expertise to the computer and to understand how a production system performs a particular task. The form of production rules also encourages the incremental growth of the rules database as knowledge of the problem environment increases (1:193).

An interpreter makes up the third component of a production system. The interpreter drives the production system by identifying the appropriate production rule to invoke from the rules database. Each production rule is compared against the local database to determine if the conditional portion of the production rule is true. When a production rule is identified with a true conditional part then the action part of the production rule is performed. The action part of the production rule modifies the local database. Once the action part has been completed, the next production rule to invoke is identified based upon the updated local database. This process continues until some terminating production rule is identified (1:190-191).

Blackboard System. The term "blackboard system" has been coined to describe a hybrid architecture for expert systems. This architecture was first introduced in 1976 with the expert system HEARSAY (1:336). This architecture calls for several expert systems, called knowledge sources, to work together on different aspects of the same problem. These knowledge sources then communicate between themselves by posting results on a global "blackboard" which each knowledge source has access to. This type of architecture has proven very successful for the design of problem solving systems (5:343-348).

Existing Expert Systems. Approximately 50 expert systems have been developed (17:60). One of the earliest expert systems, DENDRAL, was developed in the late 1960s at Stanford University. This expert system analyzes mass spectrograms and has seen use by organic chemists for over 15 years to help identify the chemical structure of organic matter (19:237-240).

The most famous expert system developed, MYCIN, was developed in the mid-1970s at Stanford University. This expert system suggests appropriate antibiotic treatment for bacterial blood infections. A physician enters the patient case history, and the system responds with a recommended treatment, why that particular treatment was recommended, and a confidence factor that the recommended treatment is correct. The diagnosis of these types of infections causes doctors considerable difficulty and MYCIN outperforms the doctors much of the time (6:40-41).

One of the first expert systems to recoup development costs was PROSPECTOR. Developed in 1976 at the Stanford Research Institute, Inc., Palo Alto, Cal., PROSPECTOR functions as a consultant to geologists. It helps assess the likelihood of a particular location containing certain ore deposits. The first time this expert system was used, it helped locate a significant ore deposit. The value of this one ore deposit more than paid the development costs of the system (2:155-162).

R1, another profitable expert system, was developed at Carnegie-Mellon University in Pittsburgh, Pa. in 1979. R1 configures VAX computers for the Digital Equipment Corp. (DEC) based upon the customer order. Previously, only those experts within DEC were capable of configuring a computer. And they often made mistakes resulting in the delivery of computers with incompatable components. R1 now configures all new computers for DEC and performs more consistently then the human experts (6:41). The success of R1 has caused most other computer manufacturers to begin developing similiar systems.

Knowledge Representation

Just as AI based programs solve problems differently than standard computer programs, AI based programs often represent knowledge about the problem differently. In fact, knowledge representation is one of the most active research areas in AI. Knowledge can exist in one of two basic forms. Declarative knowledge is knowledge as to what is known to be true or false. Procedural knowledge is knowledge on how to do things. Both forms of knowledge are necessary for the solution of difficult problems.

Frame Systems. A frame system is a knowledge representation technique which supports both forms of knowledge. The system consists of a collection of data structures known as frames to represent knowledge. Each frame consists of a collection of knowledge about a particular subject. Each piece of knowledge is contained within an entity called a slot. The slot may contain either declarative or procedural knowledge. In either case, the retrieval mechanism is the same. This means that once the knowledge is entered into the frame system, the user no longer needs to keep track of what type of knowledge is being used.

If all a frame system did was facilitate the use of both declaritive and procedural knowledge it would be a very useful knowledge representation technique. Fortunately, frame system do so much more. The real power of a frame system lies in the inheritance capabilities of the system, as well as the concepts of demons and servants.

In a frame system, a collection of knowledge about one subject (a frame) may inherit related knowledge from another frame. The knowledge in a particular frame can represent default assumptions about a particular class of objects. Then any reference to the knowledge in a frame for that type of object will inherit the default assumptions - unless the frame referenced has overriden the default assumptions. This allows each individual occurance of an object to annotate only the differences from the defaults for that object (1:216).

An example of inheritance should clarify this concept. Assume a frame with the name "ALCM" was created to contain all default knowledge about ALCM's. One of the pieces of knowledge contained in the frame specifies an ALCM has a cruising speed of 450 knots. Further, assume the frames "ALCM-1" and "ALCM-2" were created to describe two specific ALCM's. For reasons unimportant to this example, the ALCM described in the frame "ALCM-2" has a cruise speed of 500 knots. Figure No. 5 shows the hierarchical relationship which would exist between these frames. When the frame system is asked the cruising speed for the ALCM described in the frame "ALCM-1" the response will be 450 knots. Since no specific information was present in the frame "ALCM-1" on cruising speed, the cruising speed was retrieved (or inherited) from the frame "ALCM". However, the same request for the second ALCM would yield a response of 500 knots as that specific information was in the frame "ALCM-2".

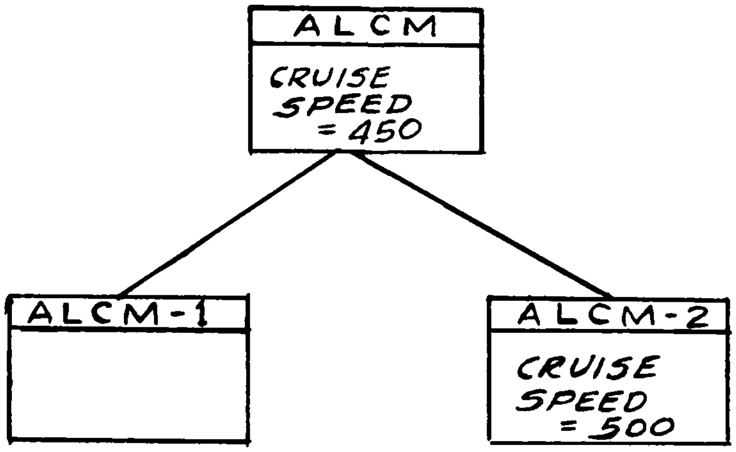


Figure No. 5 - Frame Inheritance

In addition to the inheritance capability of frame systems described above, frame systems provide the powerful concepts of demons and servants. A demon or servant is a procedure which is activated upon the occurance of some pre-designated event within a frame system. Events of interest can be described to the frame system along with the demon or servant to be executed when that event occurs. The frame system then monitors itself for the occurrance of any of the defined events. When an events occurs, the demon or servant for that event is executed. Since the user does not specifically call these procedures, demons and servants seem to just hang around the computer and do what needs to be done when necessary.

The primary difference between a demon and a servant is the nature of the procedure. From the users viewpoint, a demon tends to be restrictive in nature. However, a servant seems to expand the users capabilities. Examples should clarify the difference between the two.

Previously the frame "ALCM" has been used to contain the default knowledge about ALCMs. Assume that part of the knowledge was that minimum cruise speed is 200 knots. Let one of the defined events in the frame system be "cruise speed is less than 200 knots". Then, when the user attempts to specify a cruise speed for an ALCM of less than 200 knots the demon associated with that event will be executed. The demon may do a variety of things, but one of the key ideas is that it will not allow the user to specify a cruise speed of less than 200 knots. Often, the demon will simply notify the user that they attempted to do something that was inconsistent with the knowledge available.

However, a servant tends to expand the user capability. Assume that part of the knowledge contained in the frame "ALCM" was that a list of all ALCMs defined to the system was to be kept in the frame "ALCM". Let one of the defined events in the frame system be the event "ALCM defined". Then when the user defines a new ALCM to the frame system, the servant associated with that event will be executed. This servant could ensure that the list of ALCMs in the frame "ALCM" is updated to include the newly defined ALCM.

The specific events which may be monitored for are a function of the specific frame system being used. However, this ability to monitor the frames system for the occurance of events provides considerable power when representing knowledge.

Search Techniques

One of the most common actions necessary in problem solving is the searching of a set of alternatives in order to find the best alternative available. Because of this, search techniques have been carefully studied within the AI community. While various search techniques have been developed, only branch-and-bound techniques are discussed here because it is the technique used in this thesis. For a full treatment of search techniques the reader is referred to chapter two of The Handbook of Artificial Intelligence (1).

When discussing search techniques, it is convenient to consider a graph with one starting node (typically labeled S), several intermediate nodes, and one or more finishing nodes (typically labeled F). The links connecting the nodes are labeled with a number indicating the desirability or cost of selecting that alternative. The best alternative is that path from the S node to any F node which has the lowest cummulative cost associated with the links. Thus, the path S-A-F is the best path in Figure No. 6 with a cost of six.

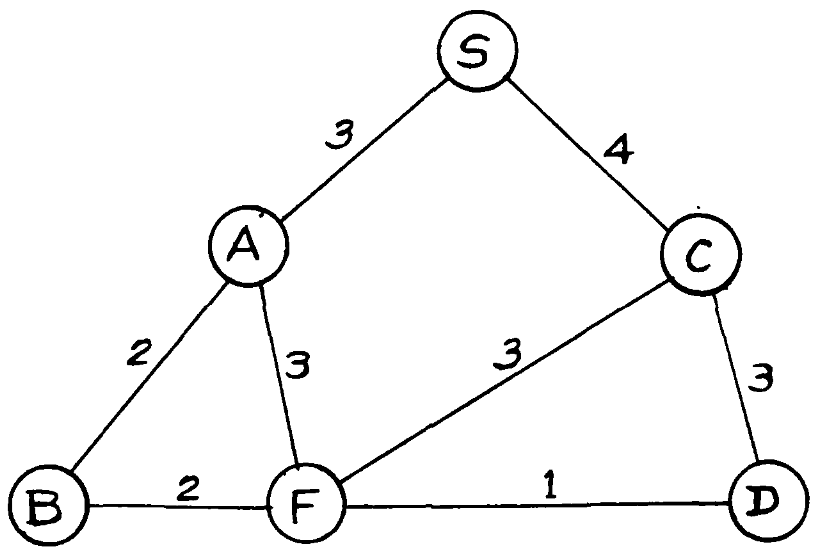


Figure No. 6 - Typical Graph

Branch-and-Bound Search Technique. The principle of a branch-and-bound search technique is that there is no need to further consider an alternative once the cost of that alternative exceeds the cost of another alternative already considered. For example, consider the situation represented in Figure No. 6. Once the path S-A-F has been considered and a cost of six determined there is no reason to consider any of the alternatives which use node C. This is because the cost associated with C is greater than the known cost of an acceptable alternative. Therefore, neither path S-C-F or S-C-D-F must be evaluated past node C before they can be eliminated from consideration. As can be seen, using the branch-and-bound technique allows a search for the optimum solution to eliminate alternatives without fully evaluating all the alternatives. And while it will not be proven here, a branch-and-bound search is guaranteed to find the optimal alternative (19:99).

The effectiveness of the branch-and-bound search technique varies with the situation. In the worst case, where each alternative considered is slightly better than the previous alternative, all alternatives will be fully evaluated before the optimum solution is found. Even in the best case situation, where the first alternative considered was the optimum alternative, most of the other alternatives will require partial evaluation before they can be eliminated from further consideration. In an attempt to improve the efficiency of the branch-and-bound technique, a modification of this technique has been developed known as the heuristic or A* search technique.

Heuristic Search Technique. It has been discovered that the performance of a branch-and-bound search technique can be improved by estimating, with a heuristic, a cost for the various alternatives. Then the lowest cost alternative, based upon the estimates, is evaluated to determine the true cost of that optimum alternative. Naturally, the success of this approach is dependent upon the accuracy of the heuristic used to estimate the costs.

The principle behind the heuristic search technique is that the heuristic used to estimate the cost of an alternative will be computationally less expensive than fully evaluating that alternative. Therefore, the key to success with a heuristic search is the identification of a good heuristic to use in estimating costs. This heuristic should require a minimal amount of computation to derive and accurately reflect the true cost of all alternatives. Typically, the heuristic used is derived by taking advantage of knowledge about the particular problem domain. As such, a heuristic which works well for one problem domain may not work at all for another problem domain.

Considering a few heuristics that could be used with the graph shown in Figure No. 6 should demonstrate the principles of a heuristic search. For each heuristic considered, let it be assumed that the costs associated with each link must be calculated in some manner and are not simply stored in some location. For the first example, let the heuristic be the sum of the path costs from the current node to the finish node. With this heuristic, the search reverts to a standard branch-and-bound search. As such, finding the optimal alternative is guaranteed. But, there will be no savings in the computational costs of finding that optimal alternative.

A very attractive heuristic for this graph would be the number of links between the current node and the finish node. This heuristic would be inexpensive to calculate and provides an excellent estimate of the true cost of a path. As can be seen by looking at Figure No. 6, the paths S-A-F and S-C-F have a length of two links. All other paths have a length of three links. Therefore, by applying this simple heuristic, all but two alternatives were eliminated without evaluating any of the alternatives.

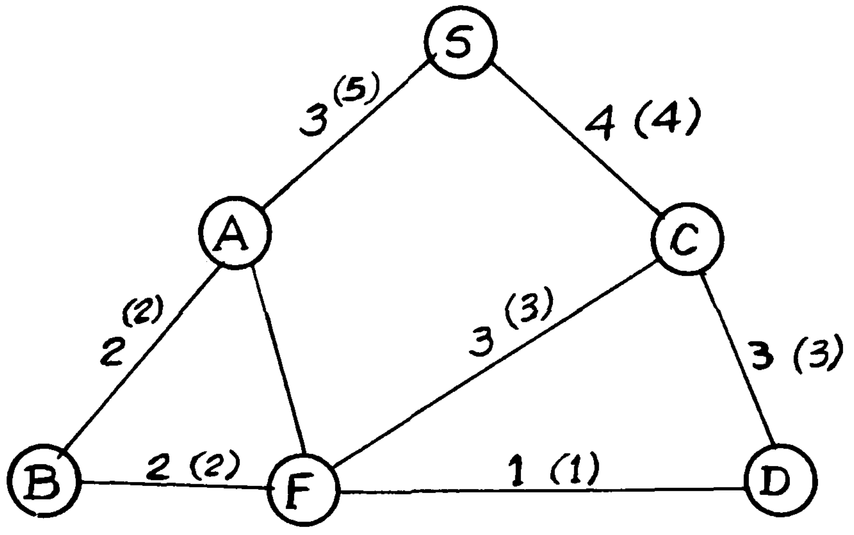


Figure No. 7 - Typical Graph With Estimates

A different heuristic, one which doesn't work, is demonstrated in Figure No. 7 - a reproduction of the graph in Figure No. 6. The costs for each link, as estimated by the poor heuristic, are within parentheses. Using these estimates, the path S-C-F would incorrectly be identified as the optimum alternative. Notice that the heuristic properly estimated the cost of every link except one. This single flaw in the heuristic resulted in the search incorrectly identifying the optimum alternative.

As can be seen, a heuristic search is precisely as good as the heuristic used. However, a heuristic search is guaranteed to find the optimal alternative if the heuristic used never overestimates the cost of a link (19:101). Thus when evaluating a heuristic search the heuristic used must be carefully examined to ensure it never overestimates the true cost of an alternative.

SUMMARY

The single most important feature of a program which identifies it as AI based is the use of symbolic processing. However, there are several other key indicators as to rather the program is AI based or not. Important among these would be the seperation of the inference engine from the knowledge base and the use of advanced knowledge representation techniques such as a frame system.

Numerous search techniques have been studied within the AI laboratories. The branch-and-bound technique is guaranteed to find the optimum solution. But, a heuristic search can usually find the solution faster. However, a heuristic search is not guaranteed to find the optimal solution unless the heuristic always underestimates the true cost of alternatives. Therefore, a heuristic search must be used with caution.

IV. Planning

Planning is the process of developing a sequence of actions to achieve a goal. It is part of the common-sense type of reasoning that people do regularly (18:269). Yet, despite this fact, planning remains a difficult task for many people and an extremely difficult domain for computer applications.

Background

When planning, it is convenient to think of each unique set of facts about the problem domain as a "state". The set of facts that are true when the planning process begins is called the "initial state". Each set of facts that are true when the planning process successfully completes is called a "goal state". Some problem domains may have multiple goal states. All other states are called "intermediate states". Additionally, the state represented by the current set of facts is also known as the "current state" (19:130).

Any function available within the problem domain which results in the facts about the problem domain changing is called an "operator". With these terms defined, the planning process can be defined as the selection of operators which allows the "current state" to change from the "initial state" to the "goal state". Depending upon the type of operators available, "intermediate states" may need to be entered before the "goal state" can be obtained.

A simple example should clarify this definition. Let the variable "I" be the only fact within a problem domain. This variable is restricted by the problem domain to being a non-negative integer less than seven. Thus, there are seven possible states within this problem domain. The planning problem to be solved for this example is "How to make the variable I equal to four given that I is currently zero?". Based upon this problem statement, the initial state is when the variable I is zero. The goal state is when I is four.

Defined within this problem domain are two operators. The first operator, called "Plus-1", will increment by one the variable I. The second operator, called "Plus-2" will increment the variable I by two. This entire problem domain can now be represented by the graph in Figure No. 8. Each state within the problem domain is represented by a node in the graph. Each arc leaving a node represents an operator that can be applied to that state; resulting in a new state.

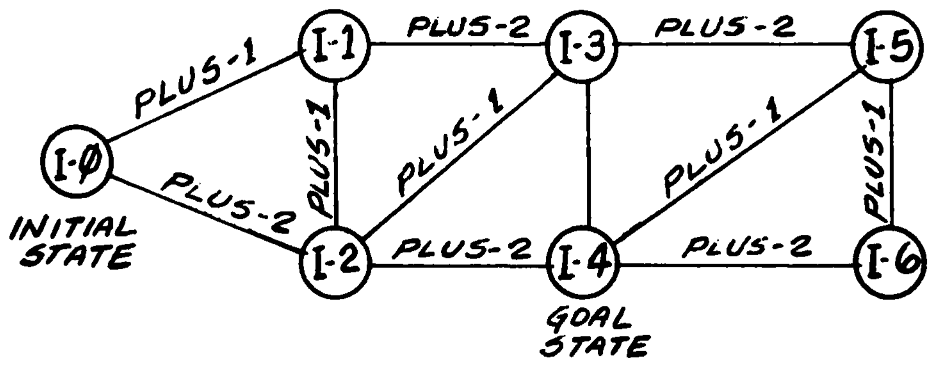


Figure No. 8 - Graph of Problem Domain

By following the flow of the graph, it becomes easy to solve this planning problem. Starting at the initial state, applying the Plus-2 operator changes the current state to the intermediate state represented by the node I=2. From this node, applying the Plus-2 operator changes the current state to the state represented by the node I=4. Since this is the goal state, the plan necessary to solve this problem is to apply the Plus-2 operator twice. Note that this is not the only combination of operators that would reach the goal state. For example, applying the operator Plus-2 once followed by the operator Plus-1 twice will also reach the goal state. Which plan is preferable depends upon the definition of best within the problem domain.

In the previous example, both operators were allowed to be applied at all states. However, often operators are subject to preconditions and constraints which may disallow the use of an operator at some states. A precondition is a set of facts which must be true or the operator may not be applied to a particular state. A constraint is a set of facts which must be true after the operator is applied. If the constraint cannot be satisfied the operator cannot be used.

As was hopefully demonstrated by the previous example, planning can be reduced to a graph search type of problem once the states and operators defined within the problem domain have been identified. There are two basic approaches to planning, hierarchical and non-hierarchical, and both are based upon this principle.

Nonhierarchical Planning Approach

Most early planning systems used a nonhierarchical approach to planning. This approach is identified by the use of a single level of abstraction and only one representation of the plan. That plan representation is the list of operators necessary to go from the initial state to the goal state. The previous example on planning is typical of a nonhierarchical planning system because only one level of abstraction was used. Some of the better known nonhierarchical planners include STRIPS, HACKER, and INTERPLAN. These are described in chapter 15 of The Handbook of Artificial Intelligence (3:523-540).

The major disadvantage of nonhierarchical planners is that they expend considerable resources on details not critical to the success of the plan. As a result, they often become bogged down in unimportant details (3:517).

Nonhierarchical Approach To ALCM Planning.

It is interesting to note that the SCT approach to ALCM planning discussed in chapter one was a nonhierarchical approach. To show this, their approach will be mapped into the vocabulary used here to describe planning.

The variables "latitude" and "longtitude" were used to define the states of their system. These variables were subject to the constraint that the difference in latitude and longtitude between two adjacent states must be equivalent to five nautical miles. They were also constrained to remain within a predefined range known as the planning area.

The initial state was that state represented by the coordinates, expressed as latitude and longtitude, of the ALCM launch point. The state represented by the coordinates of the ALCM's target was the goal state. The current state is that state represented by the coordinates of the ALCM's current position.

Eight operators were defined by SCT to be applied at each state. Each operator caused a change in the current state by changing the latitude and longtitude of the ALCM's position. The difference between operators was the heading flown by the ALCM to change position. The first operator used a heading of 45 degrees. Each subsequent operator incremented the heading by 45 degrees. Thus, the eighth operator used a heading of 360 degrees. Each operator was subject to the constraint that applying the operator would not result in a state which was off the planning area.

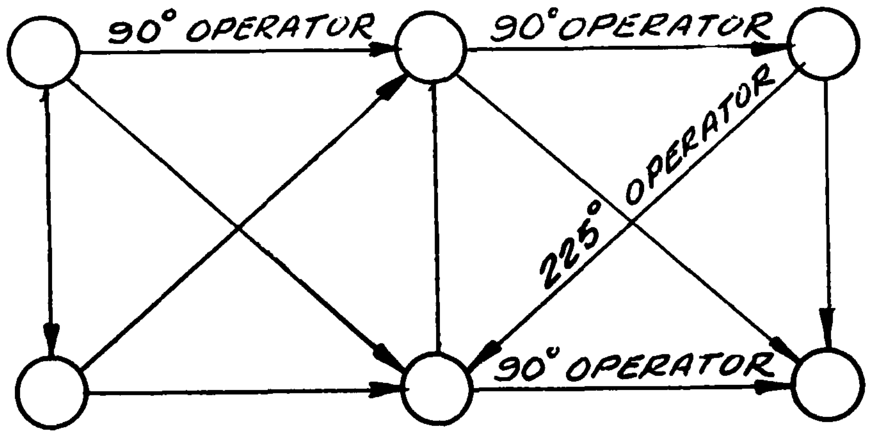


Figure No. 9 - SCT Planning Domain

The graph in Figure No. 9 shows six states from the problem domain just described. By comparing this graph with Figure No. 1 in chapter one, it becomes clear that SCT used a nonhierarchical approach. This makes it easy to evaluate the suitability of nonhierarchical planning for ALCM mission planning. Earlier, in chapter one, it was pointed out that the computational costs of the SCT approach may be prohibitively high. This is consistant with the problems of a nonhierarchical approach - becoming bogged down in unimportant details. Therefore, based upon the high computational costs of the SCT effort it can be concluded that nonhierarchical planning is not well suited to ALCM mission planning.

Hierarchical Planning Approach

Most of the more recent planning systems have used a hierarchical approach to planning. This approach is identified by multiple levels of abstraction with a seperate representation of the plan for each level of abstraction. Hierarchical planning is defined as:

> planning abstractly using a simplified model of the problem. A hierarchical planner suppresses the details of the problem in order to focus on the most important considerations (13:9).

Thus a hierarchical planner begins the planning process with a high level plan called an abstract plan. This abstract plan focuses on the more critical elements of the problem domain. Once the abstract plan is completed, the promising alternatives are replanned at lower levels of abstraction. The lower the level of abstraction, the more detail considered in the plan at that level of abstraction. The lowest level of abstraction considers all the details necessary for the plan to succeed.

The planning systems ABSTRIPS and NOAH are the classic examples of hierarchical planners. Both are described in chapter 15 of The Handbook of Artificial Intelligence (3). Of particular interest is ABSTRIPS. This planner was developed for the same problem domain as the STRIPS planning system - which was a nonhierarchical planner. Therefore, comparing STRIPS with ABSTRIPS provides a valid comparison of the two approaches to planning. (3:523-530).

The abstraction used in a hierarchical planner can be abstractions of either the operators or states defined in the problem domain. In fact, in some hierarchical planners, abstractions of both are used. For example, an ALCM mission planning system might use two levels of abstraction. At the higher level, the state space could consist of only those coordinates containing enemy defenses and TERCOM maps. The operators may allow heading changes in only 45 degree increments. Both are very high level abstractions of the true problem domain for mission planning. Once a plan is developed at this higher level of abstraction, the details of the plan can be worked out at a lower level. At the lower level of abstraction, a new state may be defined every mile and the operators may consist of any change in heading or speed.

The power of hierarchical planning comes from the ability to eliminate possible alternatives from consideration during planning at one of the higher levels of abstraction. Then those alternatives need not be considered at the lower abstraction levels and the planner does not become bogged down with details which don't contribute to the solution of the problem. Therefore, the higher the level of abstraction at the upper level; the more powerful the planner will be.

Hierarchical Planning Styles. The more important considerations used in the upper abstractions of a hierarchical planner are simply generalized constraints which can apply to all alternatives at that level of abstraction. The more general these constraints, the more powerful the planner will be. This has resulted in two different styles of hierarchical planning - heuristic and least commitment.

Heuristic Planning Style. A heuristic planning style uses highly generalized constraints called heuristics. These heuristics are rules-of-thumb which generally hold true. They allow the elimination of large numbers of alternatives within a couple levels of abstraction. This results in a fast hierarchical planner. This appears desirable, but the use of over generalized heuristics can result in the best alternative being eliminated from consideration. When this happens, either the best alternative will not be found or the hierarchical planner must be capable of identifying when this situation occurs and backtracking to the point where the alternative was eliminated.

Least-Commitment Planning Style. The potential of inadvertently discarding the best alternative has resulted in the developmemt of a least-commitment planning style. With this style of planning, only constraints (heuristics) which guarantee the best alternative will not be eliminated are used. This planning style is attractive as it eliminates the need for backtracking. However it does so at the cost of considering more alternatives. Using a heuristic planning style is faster; but, a least-commitment style guarantees the best solution.

<u>Hierarchical Approach To ALCM Planning</u>. Planning ALCM missions appears to be a problem domain well suited to a hierarchical planning approach. This can be demonstrated by considering the situation depicted in Figure No. 10. Using only two levels of abstraction, a mission plan can be efficiently developed for the sortie.

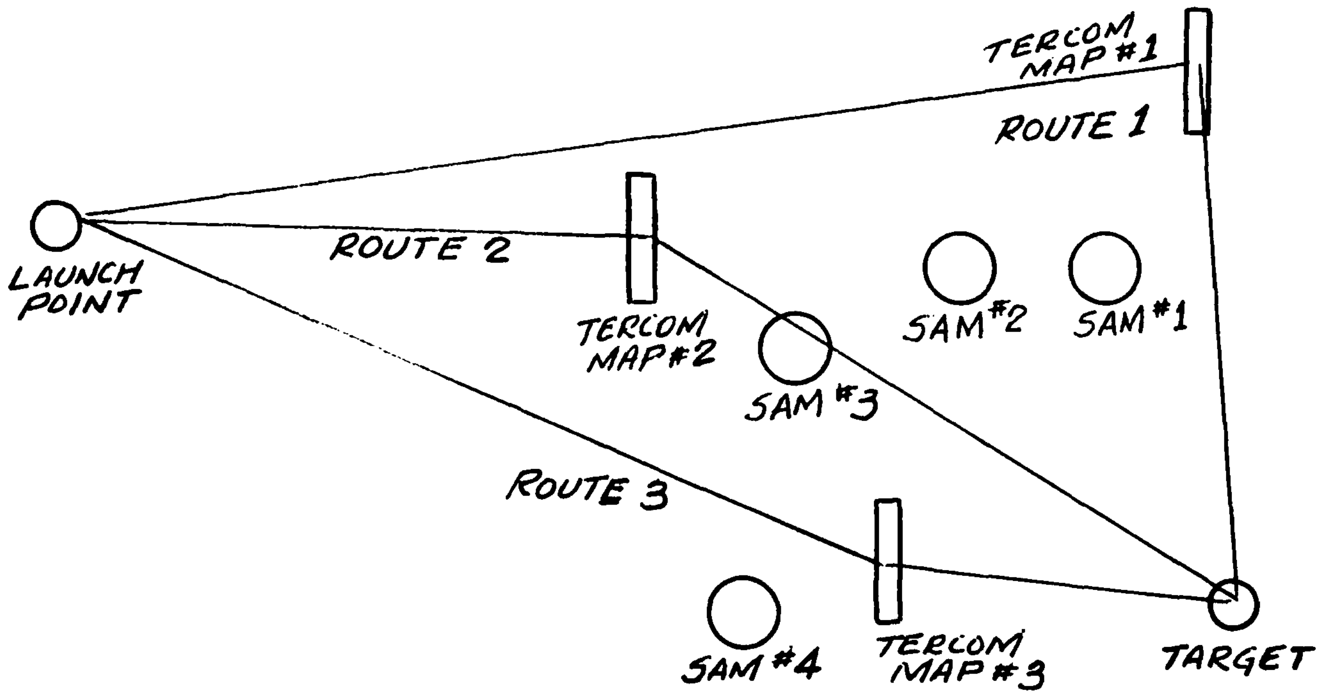


Figure No. 10 - Sample ALCM Mission Scenario

For the upper level abstraction, let the state space consist of the set of coordinates (rounded to the closest tenth of a degree) for the launch point, target and all TERCOM maps. Let the operators consist of the set of 360 heading changes from a state with one degree increments. These operators are subject to the following constraints:

1) If the current state is the coordinates of the launch point then movement on the heading of the operator must lead to a state representing a TERCOM map.
2) If the current state is the coordinates of a TERCOM map then movement on the heading of the operator will lead to the state representing the target.

With these operators, only the three routes shown on Figure No. 10 would be generated at this level of abstraction.

For the lower level abstraction, let the state space consist of the set of coordinates (rounded to the closest hundredth of a degree) for the launch point, target, all TERCOM maps, and all points within the coverage of a SAM site. The operators at this lower level are the same that were used at the higher level. However, in addition to the original constraints the following constraints are applied:

1) The cumulative distance from the initial state to the next state must be less than the ALCM range.
2) The current state, after applying an operator, must not be a state representing a point within the coverage of a SAM site.

These operators when applied to the three routes that were promising at the upper level results in only route #3 being acceptable. Route #1 is rejected due to excessive distance. Route #2 is undesirable because it intersects the coverage of SAM #3. Thus route #3 becomes the generated plan.

Notice that by delaying consideration of the SAM sites until the lower level of abstraction, the number of alternatives considered at both levels of abstraction were significantly reduced. Yet, it does not impact the suitability of the final plan. This was because only SAM #3 impacts the plan. Considering the other SAM sites simply does not contribute to solving the problem. That is the principle of hierarchical planners and it works well within a difficult problem domain like ALCM mission planning.

Meta-Planning Approach

In 1980 a variation on hierarchical planning, called meta-planning, was introduced. Meta-planning means to plan about the planning process. Therefore, the principle behind a meta-planning approach is to use a standard hierarchical approach to the planning problem. However, the planning style is allowed to be either heuristic or least-commitment. Thus, a meta-planning system evaluates the planning problem for each abstraction level and concludes which style of planning is appropriate for that abstraction level.

MOLGEN. The idea of meta-planning was introduced with the development of a planning system called MOLGEN developed by Mark Stefik (13). In the MOLGEN system, the planning process within each abstraction level was controlled in three layers called planning spaces. The top layer, called the strategy space, examined the current planning problem and selected either a heuristic or least commitment planning style based upon the situation. The middle layer, called the design space, organized possible design options based upon the planning style selected in the strategy space. The bottom layer was called the laboratory space. This bottom layer evaluated the designs proposed in the design space. The results of these evaluations were reported to the design space and used to eliminate design alternatives (13:5-6).

In addition, to using a meta-planning approach, MOLGEN was significant due to the architecture of its laboratory space. Previously, AI planning systems had been implemented as pure AI systems. Therefore, the systems religiously followed the architectural principles of expert systems discussed in chapter 3. However, in MOLGEN the laboratory space consisted of software developed in the conventional manner. Thus, the design space was an AI based system which was used to drive conventional software. An interesting twist in software development philosophy.

Summary

There are two basic approaches to planning; hierarchical and non-hierarchical. In either case, the planning problem can be reduced to a graph search type problem if the problem domain is defined in sufficient detail. Therefore, to do planning the state space and operators for the problem domain must be identified.

V. MPLANNER System Overview

The system designed in this thesis for planning ALCM missions is called MPLANNER, for Mission PLANNER. MPLANNER is based upon an AI framework as the overall structure and planning approach coming from recent AI research. However, the system is not a pure AI system as there are components designed as a mixture of conventional software and AI techniques. The system consists of four major components; the Dual Blackboard, the User Interface, the Database Manager, and the Mission Planner. Each is built upon the AI based knowledge representation frame system CP-FRL developed by by the author. These components work together to implement a controlled hierarchical approach to mission planning.

MPLANNER Architecture

In chapter three, a blackboard system was described as a set of expert systems working in conjunction with each other to solve a problem where each expert system was called a knowledge source (KS). By relaxing the definition of a KS, it can be shown a blackboard system makes an excellent architecture for an ALCM mission planning system. However, potential implementation difficulties resulted in the development of the dual blackboard system concept. This is a hybrid of a conventional blackboard system and is used as the basic architecture of MPLANNER. Therefore, MPLANNER should be called a dual blackboard system.

__Suitability of Blackboard Architecture__. For this thesis, a KS will be more loosely defined as any software, AI based or not, which uses a blackboard as it's interface with the other parts of the system. With this definition of a KS, blackboard systems could become the standard architecture for difficult problem domains in the future. There are two primary reasons why this could occur. First, blackboard systems implement several key concepts of software engineering. Second, a blackboard system can apply the power of parallel execution to the problem domain when implemented on a distributed system - even if not initially developed for a distributed system.

The cornerstone of software engineering lies in the theory of using a divide and conquer approach to difficult problems. This dictates that difficult problems be divided into managable subproblems which can be solved individually. The interface between these subproblems must be clearly defined and one subproblem should never need to understand how another subproblem is solved. This concept is known as information hiding in the software engineering field. Software maintainability is another key concept, as any system actively used will require changes. It must be possible to change the system in a reliable and cost efficient manner.

A blackboard architecture implements all these concepts so critical to the successful development of computer systems in difficult problem domains. Each KS is a clear subdivision of the total problem with the interface defined as the messages on the blackboard. Since only messages are posted on the blackboard, absolutely no information is necessary as to how each KS actually functions. And most important, new knowledge sources may be added with a minimum of effort - allowing the system to grow as it matures.

Despite the appeal of a blackboard architecture from a software engineering perspective, the real clincher for selecting a blackboard architecture when designing a new system is the role distributive processing will play in the future. Given the current level of activity in both computer networking and special purpose computer architectures, it is a fairly safe prediction that distributed computer environments will see more widespread use in the future. It is not uncommon in computer systems today to offload computationally expensive functions to peripheral processors like array processors. This is a limited form of a distributed system. As communications between computers becomes easier (more standardized) and more special purpose computer architectures, such as database machines and logic machines, become available distributive processing systems will see more widespread in use. When this occurs, a system which was designed around a blackboard architecture will be easy to implement on a distributive system.

A blackboard architecture can be viewed as a simulated distributive processing system by considering each KS as a different processor and the blackboard as the network connecting the processors. When viewed in this manner, it is easy to visualize how a system designed around a blackboard architecture can be implemented on a distributed system with few changes. Once implemented on a distributive system, the blackboard system becomes a true powerhouse. Then different knowledge sources can consider alternative approaches to the same problem with the knowledge sources executing simultaneously on different processors.

As can be seen, a blackboard architecture is an ideal architecture for difficult problem domains. It forces compliance with several of the key principles of software engineering. This enhances the probability that the system will function as expected after it has been developed. There are other design architectures that also enforce good software engineering principles, but none which allow a system to be developed on a nondistributed computer system and then at some future date be moved to a distributed system with only a few changes. Any other architecture would require a major redesign to capitalize on the parallel processing potential associated with a distributed system. This ability to take advantage of distributed systems as they become available in the future was the primary reason for selecting a blackboard architecture for MPLANNER.

Dual Blackboard Concept. Despite the attraction of a blackboard architecture, a problem does arise when a blackboard system is implemented. Each KS in a blackboard system communicates with each other by posting requests and conclusions to a central blackboard. This typically results in a large blackboard by the time all the data needed by each KS is also placed on the blackboard. A large amount of data on the blackboard usually results in the blackboard frequently being accessed and becoming a source of contention within the system. Therefore, to reduce the size (complexity) of the blackboard the dual blackboard concept was developed.

A dual blackboard system consists of both a message blackboard and a data blackboard, with each KS having access to both blackboards as shown in Figure No. 11. The message blackboard fulfills the stylistic intent of a blackboard system as the central repository of all messages necessary for the knowledge sources to work together in a coordinated manner on the problem. The messages should consist of a keyword, identifying the message type, and a few parameters. Each KS monitors the *message blackboard* for pertinent messages. When the KS identifies a applicable message, it is processed by the KS and any conclusions reached are posted back to the message blackboard.

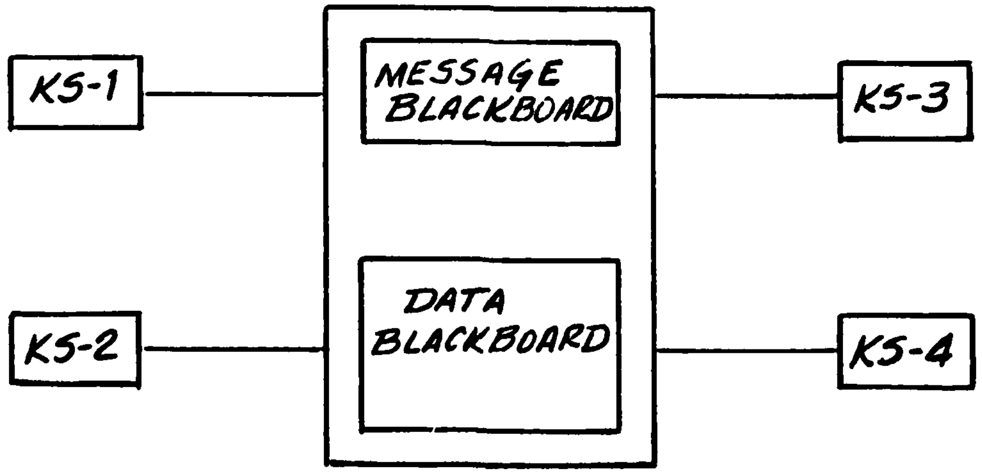


Figure No. 11 - A Dual Blackboard System

The data blackboard facilitates the sharing of data between knowledge sources and allows the message blackboard to remain uncluttered. As such, the data blackboard contains that subset of the system database necessary to solve the portion of the problem currently under consideration. Every KS may retrieve data from the data blackboard, but restrictions are placed on what data a KS may post to it. For each category of data on the data blackboard only one KS may post that category data to the blackboard. For example, in the ALCM mission planning problem domain, only one KS posts the TERCOM map data to the data blackboard. If another KS needs TERCOM map data not already on the data blackboard, it must send a message to the KS controlling the TERCOM map data requesting the needed TERCOM map data be posted to the data blackboard. While this slightly increases the traffic on the message blackboard, it does much to simplify the use of the data blackboard.

Advantages Of A Dual Blackboard System. The concept of a dual blackboard sytem has two primary advantages over a standard blackboard system. The first is a matter of conceptual clarity. The conceptual attraction of a blackboard system lies in the simple interface between each KS. Simply post a message to the blackboard and, if appropriate, watch for a response. But, any attempt to implement a blackboard system quickly clutters that clean interface with data. At some point, the blackboard becomes a global database; not a message center. When this occurs the blackboard system concept becomes destroyed by the implementation. However, the dual blackboard concept can be implemented while maintaining conceptual clarity. By providing a facility for the sharing of data between each KS, the message blackboard can remain true to the concept of blackboard systems.

The second advantage of a dual blackboard system is more practical - speed. Each KS must monitor the blackboard to determine if they can contribute to the solution of that portion of the problem currently under consideration. This means each KS must periodically examine each item posted to the blackboard to see if there are any messages which apply to the KS. As the message blackboard contains only messages and no data it can be examined for pertinent messages much faster than a standard blackboard which has data interspaced among the messages.

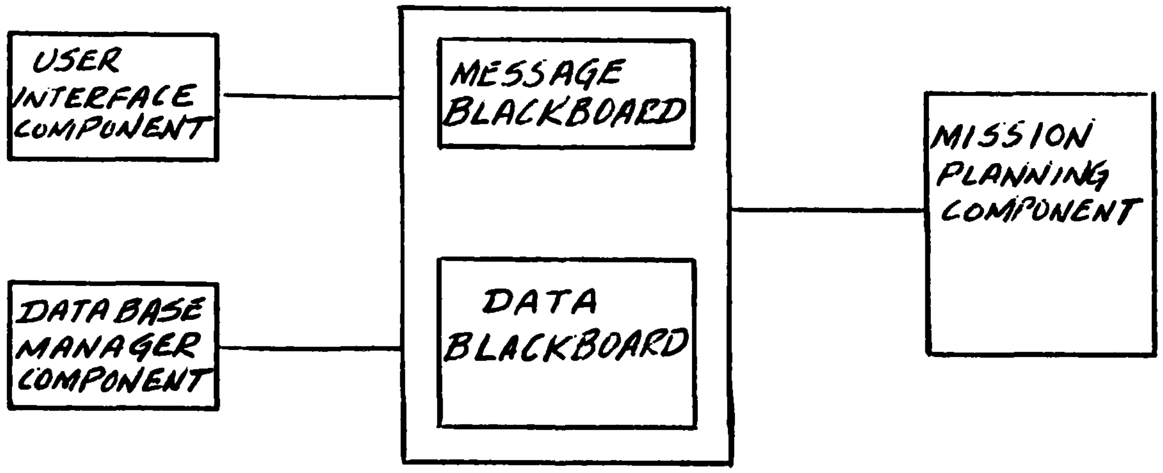


Figure No. 12 - MPLANNER Dual Blackboard Architecture

MPLANNER Design. Because of the advantages inherent in blackboard systems in general, and the implementation advantages of a dual blackboard architecture, MPLANNER has been designed as a dual blackboard system. The user interface, database manager, and mission planning components all communicate via the message and data blackboards of the dual blackboard. The dual blackboard component of MPLANNER, provides the interface between the blackboards and the other components of the MPLANNER system. Figure No. 12 shows the architectural design of the MPLANNER system.

Dual Blackboard Component

The dual blackboard component of MPLANNER is the glue which holds the system together and coordinates KS access to the blackboards. The dual blackboard component consists of a set of functions called by each KS when they need access to a blackboard. These functions ensure that only one KS has access to a particular data item at a given time. This ensures the integrity the data posted to the blackboards.

User Interface Component

The user interface component consists of one KS called the User-Interface-KS. This KS is the primary interface between the mission planner and the MPLANNER system. As such, this KS receives all user requests and determines if the request can be satisfied by MPLANNER. All requests fall into one of three categories; manipulate the database, display information, or plan a mission. When the mission planner inputs a request, the User-Interface-KS will post that request on the message blackboard. Once the appropriate KS has processed the request, the User-Interface-KS will notify the mission planner that the request was processed and wait for additional requests from the mission planner.

Database Manager Component

The database manager component of MPLANNER consists of the KS known as the Database-Manager-KS. This KS performs all the database functions required by MPLANNER. When the system is initially started, the Database-Manager-KS posts to the data blackboard a small subset of the MPLANNER database. While MPLANNER is executing, the KS satisfies all requests from the mission planner for database manipulations such as adding or deleting data. In addition, this KS posts to the data blackboard any portion of the MPLANNER database that may be requested by other knowledge sources. When the mission planner is finished with the MPLANNER system, any modifications made to the MPLANNER database will be saved to disk storage - if indicated by the mission planner.

Mission Planner Component

The mission planner component of MPLANNER is the heart of the MPLANNER system as this component performs the actual mission planning. As was discussed in earlier chapters, the major difficulty with planning is the large number of alternatives which must be considered before the optimal solution can be identified. Therefore, the goal of the mission planner component is to reduce, as much as possible, the number of alternatives which must be considered; while guaranteeing the optimal solution will not be eliminated from consideration and will in fact be identified. To accomplish this, the mission planner component is built around two basic ideas - hierarchial planning and the active use of constraints.

Planning Approach. The mission planner component implements a modified meta-planning approach to mission planning. As was discussed in chapter four, meta-planning is a hierarchical approach to planning with three distinct layers of control applied to each level of abstraction used by the planning system.

The top layer of control in a meta-planning system, called the strategy space in MOLGEN, determines the planning style (heuristic or least-committment) appropriate for that level of abstraction. However, MPLANNER does not use a top level of control because the heuristic planning style is always appropriate for this problem domain. Heuristics must be used if the number of alternatives to be considered are to be reduced. Therefore, the key to success is to find a heuristic which can eliminate alternatives without possibly eliminating the optimum solution. Fortunately, such heuristic exist for this problem domain.

The middle layer of control in a meta-planning system, called the design space in MOLGEN, organizes possible design options. Within MPLANNER, the KS named Route-Planning-KS performs these functions. For each level of abstraction the KS proposes mission plans which appear promising. If the evaluation of a proposed mission plan identifies a flaw in the mission plan then alternatives to that mission plan are suggested. Each proposed mission plan is assigned a measure of merit which reflects the probability of the ALCM reaching it's target using that mission plan. The proposed mission plan with the highest measure of merit is suggested to the mission planner as the optimum route for that sortie.

The lower layer of control in a meta-planning system, called the laboratory space in MOLGEN, evaluates the design alternatives suggested by the design space. The MPLANNER design space consists of two knowledge sources. The first, the Route-Evaluation-KS, evaluates proposed mission plans in terms of the vehicle being destroyed by enemy defenses or crashing into the ground. The Vehicle-Performance-KS evaluates proposed mission plans in terms of the vehicle performance. Can the vehicle safely fly the proposed mission plan and what impact will it have on the accuracy of the vehicle's navigation system. Both knowledge sources report to the design space components of the measure-of-merit computed for the proposed mission plan and a justification for the evaluations provided.

Since MPLANNER uses a heuristic planning style for each abstraction level it is not strictly a meta-planning system. However, META-PLANNER does use most of the control structure associated with a meta-planner. Thus, the author prefers to view MPLANNER as a controlled hierarchical planner.

Abstraction Levels. As was discussed in chapter four, a hierarchical planner plans at multiple levels of abstraction. At the upper levels of abstraction, the more important factors in the plan are considered. Each lower abstraction level refines the plan by considering additional details. The design specifications for MPLANNER call for three levels of abstraction to be used with abstraction of both the state space and the operators at each level of abstraction.

At the highest level of abstraction, only those factors dealing with the feasibility of even trying to plan the mission are considered. These include issues such as does the vehicle have enough range to reach the target and can the desired Time-Of-Arrival constraint be satisfied. The second level of abstraction considers those factors involved in the usage of TERCOM maps. While the lowest abstraction considers the factors involved in avoiding enemy defensive sites.

These abstraction levels consider only a few of the factors involved in planning realistic mission plans. Even the factors considered are not done at a sufficient level of detail for realistic mission planning. However, this design does show the approach that would need to be used in developing a production system. And it must be remembered that this is the goal of this thesis.

Constraint Usage In Mission Planning. With a little thought about the mission planning domain it becomes clear that the optimal flight path for a mission is simply the straight line from launch point to target. This flight path minimizes the duration of the flight. The longer the flight, the greater the chance of mechanical failure or encountering severe weather. Either of which reduces the chances of the mission being successfully completed.

If the optimal flight path is known then why plan the mission? Because a straight flight path does not usually satisfy the constraints placed upon it by the mission planner. Therefore, ALCM mission planning can be thought of as finding the closest approximation to a straight flight path which satisfies all the constriants imposed upon it by the mission planner.

Constraint manipulation and satisfaction is the basic premise behind the MPLANNER system. The optimal flight path is always known and at each level of abstraction additional constraints are applied to that flight path. If one of these constraints cannot be satisfied by the optimal flight path then that flight path is modified so it does satisfy the constraint. Therefore, the planning problem reduces to finding a constraint that is not satisfied and fixing up the flight path so that it can satisfy the constraint.

Knowledge Representation

The entire MPLANNER system is built upon a frame based knowledge representation system called CP-FRL, which was developed by the author. A frame system was selected because of three major features associated with frame systems. The first is the ability to retrieve knowledge in a consistent manner regardless of the form that knowledge is in - either procedural or declaritive.

The second reason was the inheritance capabilities of a frame system. It is true that any knowledge inherited can be duplicated to obtain the same result as a frame system. However, this would greatly increase the difficulty involved in implementing a large system like MPLANNER.

The third reason for selecting a frame system for knowledge representation was the utility of demons and servants. They are well suited for performing those bookkeeping functions which are so easy to forget. For example, the MPLANNER database keeps a list of all defensive sites defined in the system. A servant updates this list when appropriate. Thus the programmer is not required to remember that this list needs to be updated. Granted they should remember, but the important consideration is that the successful operation of the system is no longer dependent upon them remembering. Therefore, the more bookkeeping functions that are done with servants the less likely the system will fail at some time in the future because a programmer forgot a trivial point.

In addition to bookkeeping functions, demons and servants are well suited to enforcing constraints. They can identify when a constraint is violated and take corrective action immediately without specific action being requested. For example, a demon exists which checks to see if the MPLANNER system attempts to increase the cruise speed of an ALCM above the maximum speed of the vehicle. If this ever occurs the cruise speed is forced to the maximum speed of the vehicle and the mission planning component is advised that the vehicle is at it's maximum speed. As a result, no-where in the entire MPLANNER system is consideration given to excessive vehicle speed. The demon ensures the condition cannot occur and it becomes a detail that did not need to be considered when designing MPLANNER. Given that ALCM mission planning is the application of constraints to a flight path the use of demons and servants is very applicable in this problem domain.

Summary

MPLANNER is a dual blackboard system which is built upon a frame based knowledge representation scheme. The system consists of five knowledge sources. The interface between the mission planner and the MPLANNER system is provided by the User-Interface-KS. The Database-Manager-KS performs all the database functions required by the system. The other three knowledge sources are used to implement a controlled hierarchical approach to mission planning which has a control structure similiar to meta-planning systems.

VI. MPLANNER System Design (Part One)

In the next two chapters the design of MPLANNER is described in detail. This description is presented at a functional description level as the intent of this thesis is to suggest an appropriate design for an ALCM mission planning system - not to build such a system.

The description of the MPLANNER design begins by defining the computer system environment necessary to implement the proposed design. This is followed with a description of the overall design of MPLANNER and how the system is intended to operate. This includes how the dual blackboard was designed and the interfacing of the knowledge sources to the dual blackboard. After the system design is described, each individual KS is described in detail. The User-Interface-KS and Database-Manager-KS are described in this chapter. The other knowledge sources, making up the mission planning component of MPLANNER, are described in the next chapter.

Computer System Environment

During the process of designing software, assumptions must be made as to the type of computer system environment that will be used for implementing the software. The availability, or sometimes the non-availability, of computer resources often dictate the alternatives available during the design of software. As a result, it is important for the reader to understand the type of computer system environment that MPLANNER was designed to be implemented on. The prototype software developed for MPLANNER was implemented on a Symbolics LM-2 LISP machine. However, the design of MPLANNER was NOT tied to this machine. It is important to realize that this design can be implemented on almost any computer and does not require a LISP machine type architecture. For that matter, there is nothing inherent in the design of MPLANNER dictating the use of the programming language LISP - it was simply convenient to use it.

The design of MPLANNER is fairly robust and has only two requirements that a computer system must satisfy before MPLANNER can be implemented on that computer system. Each KS was designed to execute as a seperate process within the computer system. Therefore, the computer system must support multiprocessing. Additionally, each KS was designed to have a seperate window dedicated to that KS for displaying status information about the KS. This requires the computer system support the concept of multiple windows. Therefore, any computer system which supports both multiprocessing and multiple windows can be used to implement MPLANNER.

System Design

As was described in chapter five, MPLANNER is a dual blackboard system with five independent knowledge sources. Each KS is designed to execute as a seperate process on the computer and each KS has a dedicated window. Coordination between the knowledge sources is accomplished with the dual blackboard as each KS monitors the message blackboard for messages. While each KS performs different functions, the processes associated with the knowledge sources all function in basically the same manner.

Processes. When the MPLANNER system is started, a process is built for each KS and the process begins executing the software for it's respective KS. Each KS, except for the User-Interface-KS, function in a similiar manner. After executing an initialization routine, they place their process in a sleep mode where the KS execution is suspended. The KS then monitors the message blackboard for a message destined for that KS. When such a message is found, the process for that KS is awakened and the KS performs the function appropriate for the message. When the KS finishes with the message, the KS puts its process back to sleep and awaits receipt of another message. When a "EXIT" message is received, the KS executes a termination routine and kills it's own process.

The User-Interface-KS functions in a different manner. After executing the initialization routine, it does not put it's process to sleep. Instead, the User-Interface-KS waits for input from the mission planner. When an input request is received, a message is posted to the message blackboard identifying the mission planner request. Then the process for the User-Interface-KS is placeed in the sleep mode and begins monitoring the message blackboard. When a message is posted to the message blackboard indicating the request has been satisfied, the process for the User-Interface-KS is awakened. Once awakened, the KS waits for additional input from the mission planner.

When the mission planner requests to exit the MPLANNER system, the User-Interface-Ks places an "EXIT" message on the message blackboard and then kills it's own process. As the other knowledge sources see the "EXIT" message they will also kill their process. When all five processes have been killed the MPLANNER system has been terminated.

Dual Blackboard. As can be seen, the dual blackboard is the heart of the MPLANNER system. The message blackboard component of the dual blackboard is designed to be a single list where each item in the list is a different message. Therefore, posting a message to the message blackboard simply entails appending the message to the list of existing messages. The message blackboard structure needs to be kept simple as each KS periodically checks it for messages. The simplier the structure the faster MPLANNER will execute.

While the message blackboard is a single item, the data blackboard is a set of lists with one list for each type of data maintained in the system database. Each item of a list is the name of a data frame for the particular type of data that the list represents. This use of a seperate list for each type of data facilitates ensuring that only one KS updates a particular type of data on the data blackboard.

Windows. When the MPLANNER system is started and the processes are built, a window is also built for each KS. By default, the only window exposed (visible to the mission planner) is the window for the User-Interface-KS. The other windows are having information placed on them, but are not visible to the mission planner unless the mission planner specifically exposes them. Therefore, the mission planner does not become overwhelmed with a mass of detail on how the system is functioning. However, if the mission planner does wish to follow the planning process all they need to do is expose the windows associated with any KS of interest. This allows the mission planner to follow the planning process in detail and obtain confidence in the systems performance.

Since the User-Interface window is the only window always exposed, any KS may display a critical message on this window. This is to ensure that any critical messages will be seen by the mission planner. There are only a few of these critical messages and are used only when a KS cannot continue due to unusual situation.

The five knowledge sources which make up the MPLANNER system were each designed semi-independently of each other. The only consideration given for interaction between each KS was to ensure a consistent format for the various messages and data items on the blackboards. As a result, the design of each KS varies considerably. While all the knowledge sources use the CP-FRL frame system for knowledge representation, the Route-Planning-KS is the only KS which is clearly an AI expert system. The other knowledge sources are of a more conventional design and simply implement current AI techniques to varying degrees.

User-Interface-KS

The interface between the MPLANNER system and the mission planner, the User-Interface-KS accepts requests from the mission planner and places the appropriate messages on the message blackboard. Once the message is posted, the User-Interface-KS goes to sleep until message is received indicating the request has been handled. Then the process wakes up and waits for additional mission planner requests.

When the MPLANNER system is started, the User-Interface window associated with the User-Interface-KS is exposed with a message displayed that notifies the mission planner that the User-Interface-KS is waiting for the MPLANNER database to be initialized before continuing. Once the database has been initialized, the message is erased and the main menu for MPLANNER is exposed at the top of the screen. As viewed by the mission planner, the screen is divided into two windows. The top part of the screen contains the main menu with six options to be selected. The rest of the screen is utilized by the User-Interface window for messages to the mission planner and graphics displays. Figure No. 13 shows how the screen appears to the mission planner.

The interface between the mission planner and MPLANNER is menu driven. Thus, to enter a request the mission planner selects one of the options from the main menu. Then, based upon the selected option, additional menus may appear; allowing the mission planner to refine that request. Each of the options available are described below.

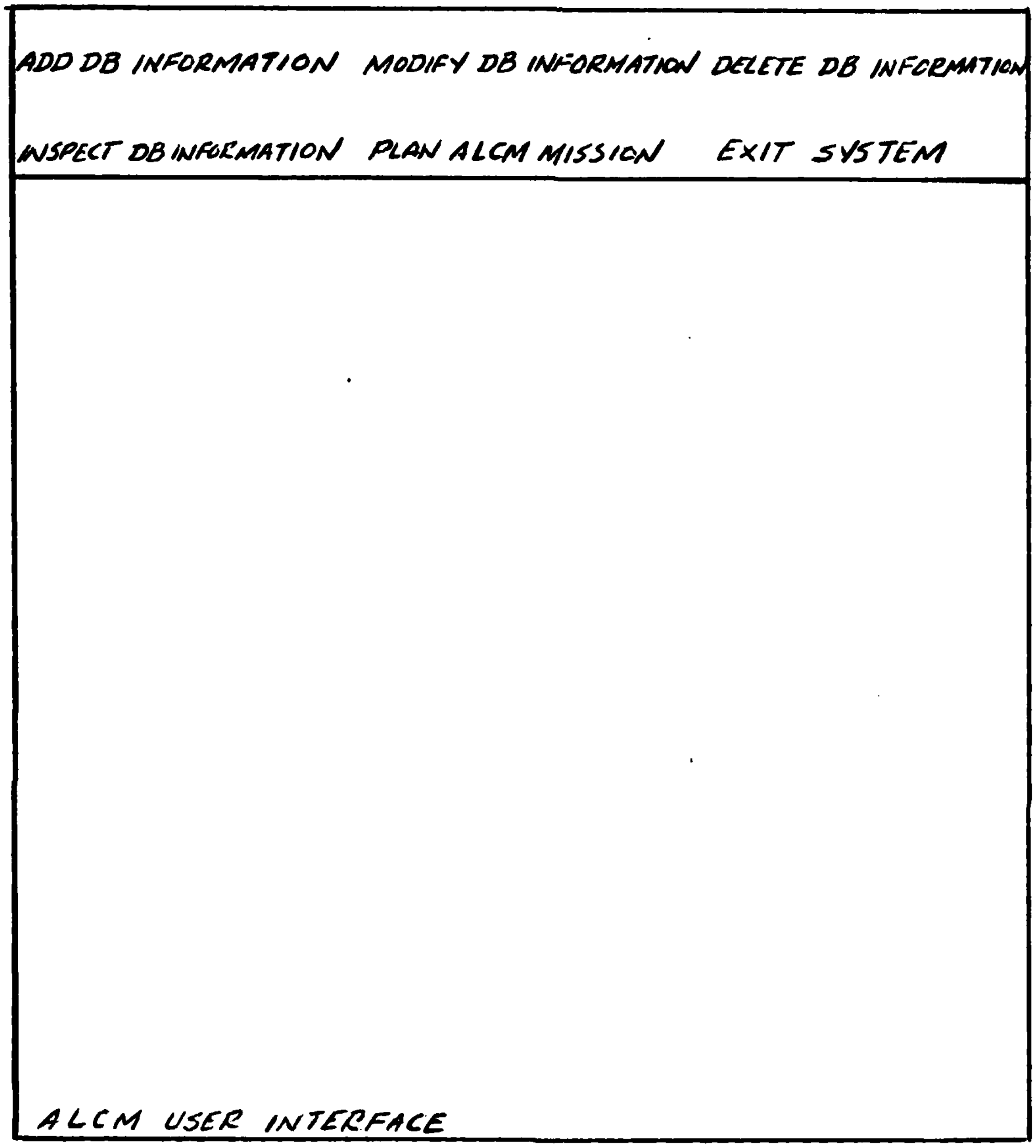


Figure No. 13 - ALCM User Interface Window And Main Menu

Add Database Information. When the mission planner wishes to add information to the MPLANNER database, they should select this option. When selected, the option displays a second menu which allows the mission planner to select the type of information to be added to the database.

Currently, the following types of information may be added to the database; Tercom Maps, Targets, and Enemy Defenses.

After the mission planner selects the type of information to be added to the database, an "ADD" message is placed on the message blackboard. This message has two elements. The first is the word "add" identifying the message type and the second element is the type of information to be added.

Once the "ADD" message has been posted to the message blackboard, the User-Interface-KS goes to sleep until an "ADD-COMPLETE" message is posted to the message blackboard. At that time, the KS process awakens and the mission planner may select another option from the main menu.

Modify Database Information. When the mission planner needs to modify information already in the MPLANNER database, this option should be selected. With this option, the mission planner can modify only those types of database information which a mission planner may add to the database. Any information maintained by the MPLANNER system may not be modified by the mission planner.

As with the add option, when the modify option of the main menu is selected a second menu is displayed which allows the mission planner to select the type of information to be modified. After selecting the type of information to be modified, a third menu is displayed. This menu contains a list of all the data frames defined in the MPLANNER database of the type selected by the mission planner. From this menu the the specific data frame to be modified is selected.

Once the data frame is selected, a "MODIFY" message is placed on the message blackboard. This message consists of three elements; the word "modify", the type of data to be modified, and the name of the data frame to be modified. The User-Interface-KS then goes to sleep awaiting a "MODIFY-COMPLETE" message. When this message is received, the KS will wake up and accept more input from the mission planner.

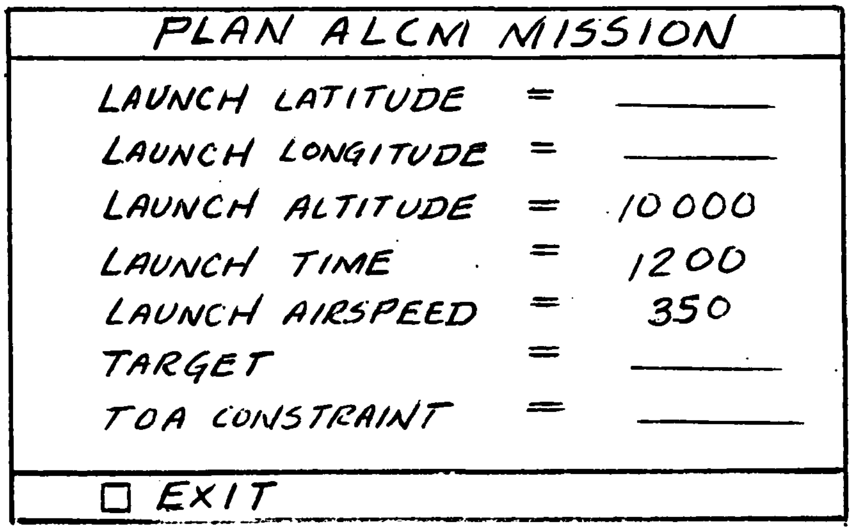
__Delete Database Information__. This option should be selected when the mission planner desires to delete existing information from the MPLANNER database. When selected, a second menu is displayed from which the mission planner specifies the type of information to be deleted. Currently, the following types of information may be deleted; Tercom Maps, Targets, Enemy Defenses, and ALCM Sorties.

Once the type of information to be deleted is selected, a menu of all data frames of that type is displayed and the mission planner selects the specific data frame they wish to delete. After the data frame is selected, a "DELETE" message is posted to the message blackboard. The message consists of three elements; the word "delete", the data type, and the name of the data frame to be deleted. The KS then goes to sleep until a "DELETE-COMPLETE" message is posted to the message blackboard - at which time it resumes accepting mission planner requests.

__Inspect Database Information__. When a mission planner wishes to inspect information in the MPLANNER database, this option should be selected. While the modify option allows the mission planner to modify only user specified elements of the database, this option allows the mission planner to view both user specified and system maintained elements. Like the modify and delete options, this option displays a menu of possible information types when selected. The following types of information can be inspected: Area Maps, Tercom Maps, Targets, Defense Sites, and ALCM Sorties.

After the type of information to be inspected is selected, a menu of all data frames defined in the MPLANNER database, of that type, is displayed. Using this menu, the mission planner selects the data frame to be inspected. An "INSPECT" message is then posted to the message blackboard. This message has three elements; the word "inspect", type of information to be inspected, and the name of the data frame to be inspected. Once the message is posted to the message blackboard, the User-Interface-KS goes to sleep until a "INSPECT-COMPLETE" message is posted to the blackboard.

Plan An ALCM Mission. To plan an ALCM mission, the mission planner would select this option. When selected, this option exposes a window on top of the MPLANNER main menu. This window is used by the mission planner to specify the parameters necessary to plan an ALCM mission. These parameters include the launch point coordinates of the ALCM and it's flight attitude (heading, altitude, and airspeed) when launched. The mission planner also specifies the taret for the sortie and any time-of-arrival (TOA) constraint. If a TOA constraint is specified, the mission plan proposed by MPLANNER will ensure the ALCM reaches the target before the time specified as the TOA constraint. Figure No. 14 shows what this window looks like.



PLAN ALCM MISSION

LAUNCH LATITUDE	=	______
LAUNCH LONGITUDE	=	______
LAUNCH ALTITUDE	=	10000
LAUNCH TIME	=	1200
LAUNCH AIRSPEED	=	350
TARGET	=	______
TOA CONSTRAINT	=	______

☐ EXIT

Figure No. 14 - Mission Planning Pop-Up Window

Once the mission planner has supplied the necessary information, the window is deexposed (removed from the view of the mission planner) and a "PLAN-MISSION" message is posted to the message blackboard. This message consists of nine elements. The first is the word "plan-mission" followed by these elements; launch-latitude, launch-longtitude, launch-altitude, launch-time, launch-airspeed, target-name, and the TOA constraint. After the "PLAN-MISSION" message has been posted to the blackboard, the User-Interface-KS is placed in the sleep mode until a "PLAN-COMPLETE" or "PLAN-ABORTED" message is posted to the message blackboard. Either message results in the mission planner being advised of the success of the mission planning effort. Then the User-Interface-KS waits for additional requests from the mission planner.

The "PLAN-COMPLETE" message consists of the word "PLAN-COMPLETE" and the measure-of-merit computed for the mission plan generated. Upon receipt of this message the mission planner is notified by the User-Interface-KS that a mission plan was successfully developed for the sortie and the measure-of-merit associated with the mission plan.

The "PLAN-ABORTED" message consists of the word "PLAN-ABORTED" ana a reason why the planning process was aborted. This can occur if the mission planner specified a target which was not in the MPLANNER database. If the specified target was valid, then an absolute constraint that was placed on the mission could not be satisfied. Regardless of the reason, the mission planner is advised that a mission plan could not be developed for the sortie and why.

<u>Exit The ALCM Planning System</u>. When the mission planner is finished with the MPLANNER system, this option should be selected to ensure the system terminates in a graceful manner. When the option is selected the User-Interface-KS posts a "EXIT" message to the message blackboard. This message consists of the single work "exit". After posting the "EXIT" message, the User-Interface-KS goes to sleep until a "DATABASE-WRAPUP-DONE" message is posted to the message blackboard. This is done to ensure the MPLANNER database is properly saved before the system is terminated. Once the message is received, the User-Interface window is killed and the User-Interface-KS terminated - returning the mission planner back to the computers operating system.

Database-Manager-KS

The Database-Manager-KS provides the database functions for the MPLANNER system. As such, this is the only KS in the system which has any knowledge as to the file structure of the database. Additionally, the Database-Manager-KS maintains the data blackboard for all types of data except the route-segment frames used by the mission planning components of the system. When another KS requires a data frame which is not posted to the data blackboard a message is sent to the Database-Manager-KS requesting the desired data frame be retrieved from the file system and posted to the data blackboard.

When the MPLANNER system is started, the initialization routine for the Database-Manager-KS initializes the data blackboard. For each type of data in the database, a prototype data frame is posted to the data blackboard. Each of the prototype data frames contains a list of all data frames of that type defined in the MPLANNER database. The following types of data are defined in the MPLANNER database; Area Maps, TERCOM maps, Targets, Defense Sites, and ALCM Sorties. Therefore, five prototype data frames are posted to the data blackboard during initialization. Once the prototype data frames are posted to the data blackboard, a "DATABASE-INITIALIZED" message is posted to the message blackboard. The User-Interface-KS waits for this message before displaying the MPLANNER main menu and accepting any mission planner input.

Once the initialization routine is complete the process for the Database-Manager-KS is put to sleep until a message for the KS is found on the message blackboard. There are seven messages which the Database-Manager-KS monitors the message blackboard for.

Add Message. The "ADD" message is placed on the message blackboard by the User-Interface-KS when the mission planner desires to add information to the system database. Upon receipt of this message the Database-Manager-KS exposes a window on top of the User-Input window. This window is used by the mission planner to specify the information to be added to the database. The information specified depends upon the type of data frame being added to the database.

When adding a defensive site to the database, the mission planner must provide the following information for the site; site name, defense type, maximum threat radius, and probability of the site destroying an ALCM which enters the threat radius of the site. Once the mission planner enters this information, the Database-Manager-KS posts a defense frame, under the name specified, to the data blackboard.

When adding a target to the database, the mission planner need specify only a target name and the coordinates of the target location. After this information is provided, a target frame is posted to the data blackboard under the target name provided.

When adding a TERCOM map to the database, the mission planner must provide a name for the TERCOM map, a set of three coordinates used to define the location of the map, and the probability of an ALCM obtaining a successful navigation update when overflying that TERCOM map. The set of three coordinates consists of the coordinates for three of the TERCOM map corners - the northwest, southwest, and northeast corners. Once the mission planner provides this information, a TERCOM frame is posted to the data blackboard under the TERCOM map name specified.

Once the appropriate data frame is posted to the data blackboard, that data frame is marked as needing to be saved to disk. The the Database-Manager-KS posts a "ADD-COMPLETE" message to the message blackboard and goes to sleep until the next message is found.

Delete Message. The "DELETE" message is placed on the message blackboard by the User-Interface-KS when the mission planner wishes to delete information from the MPLANNER database. When the Database-Manager-KS receives this message the data frame to be deleted is removed from the data blackboard and the file containing that data frame is erased from disk storage. The Database-Manager-KS then posts the message "DELETE-COMPLETE" to the message blackboard and goes to sleep until the next message arrives.

Exit Message. When the mission planner is finished using MPLANNER and requests to exit the system the User-Interface-KS posts the "EXIT" message to the message blackboard. When the Database-Manager-KS receives the "EXIT" message, all data frames that were created or modified during the MPLANNER session are saved to disk storage. When this is complete a "DATABASE-WRAPUP-DONE" message is posted to the message blackboard and the Database-Manager-KS process is killed.

Inspect Message. The User-Interface-KS posts to the message blackboard the "INSPECT" message when the mission planner requests to view portions of the MPLANNER database. When the Database-Manager-KS identifies the message the data frame to be inspected is printed to the User Interface window which is always exposed. After printing the data frame, a "INSPECT-COMPLETE" message is posted to the message blackboard and the Database-Manager-KS goes back to sleep.

Load Message. When any of the knowledge sources that make-up the mission planning component of MPLANNER needs a data frame posted to the data blackboard they post a "LOAD" message to the message blackboard. Upon receipt of this message, the Database-Manager-KS retrieves the requested data frame from disk storage and posts it to the data blackboard. Then the Database-Manager-KS posts a "LOAD-COMPLETE" message to the message blackboard and goes back to sleep.

Modify Message. The "MODIFY" message is posted to the message blackboard by the User-Interface-KS when the mission planner requests to modify part of the MPLANNER database. When the Database-Manager-KS sees this message, the window used for the "ADD" message is exposed with the information from the requested data frame displayed in the window. The mission planner can then change any of the information in the window. When the mission planner is finished, all of the changes made to the information in the window are made to the data frame and the data frame is marked as having been modified. After marking the data frame, the message "MODIFY-COMPLETE" is posted to the message database before the Database-Manager-KS goes back to sleep.

Plan-Mission Message. When the mission planner requests a mission plan be developed, the User-Interface-KS posts the "PLAN-MISSION" message to the message database. Upon receipt of this message, the Database-Manager-KS searches the MPLANNER database for the data frame of the sortie's target. If the target has been defined in the MPLANNER database, the data frame for that target is posted to the data blackboard and a "PLAN-SORTIE" message is posted to the message blackboard before the Database-Manager-KS goes back to sleep.

However, if the sortie's target has not yet been defined in the MPLANNER database, a "PLAN-ABORTED" messages is posted to the message blackboard with the second element of the message indicating that the target specified has not been defined yet. After posting the message, the Database-Manager-KS goes to sleep and awaits the next message.

As can be seen, the Database-Manager-KS functions in a very simple manner. Once the data blackboard is initialized the process for the KS goes to sleep and waits for an appropriate message to be posted to the message blackboard. When a message is found the Database-Manager-KS performs the function requested by the message and then goes back to sleep. When the "EXIT" message is received the database is saved to disk storage and the Database-Manager-KS process is killed.

Summary

The design of MPLANNER has been predicated upon the availability of a computer system which supports both multiple processes and multiple windows. Given this type of computer environment, the dual blackboard architecture of MPLANNER can be implemented. The first two knowledge sources were described in this chapter and it should have been noted that these two knowledge sources account for almost all of the traffic on MPLANNER's message blackboard. And yet, no parallel processing is performed by these knowledge sources. This is because the functions performed by these knowledge sources are not amenable to parallel processing. However, the mission planning component of MPLANNER described in the next chapter is amenable to parallel processing; and will justify the use of a dual blackboard architecture for MPLANNER.

VII. MPLANNER System Design (Part Two)

The previous chapter described the overall operation of MPLANNER and two of the major components of the system - the user interface and the database manager. The last component of the system, the mission planning component, is described in this chapter. This component performs the actual ALCM mission planning and is the centerpiece of the MPLANNER system. Consisting of the knowledge sources Route-Planning-KS, Route-Evaluation-KS, and Vehicle-Performance-KS the mission planning component implements a hierarchical approach to planning. However, before describing the specific knowledge sources, the approach used by MPLANNER for mission planning is discussed in detail.

Mission Planning By People

During the initial phases of the MPLANNER design, the author spent a considerable amount of time poured over a map preparing fake ALCM mission plans. The result was an understanding of how people would approach the planning problem in this particular domain.

The initial step in planning a mission is to propose an initial flight path. As was discussed in chapter five, the direct route from the launch point to target is the optimal flight path. Therefore, this is the logical flight path to initially propose and is represented by a dotted line in the example in Figure No. 15.

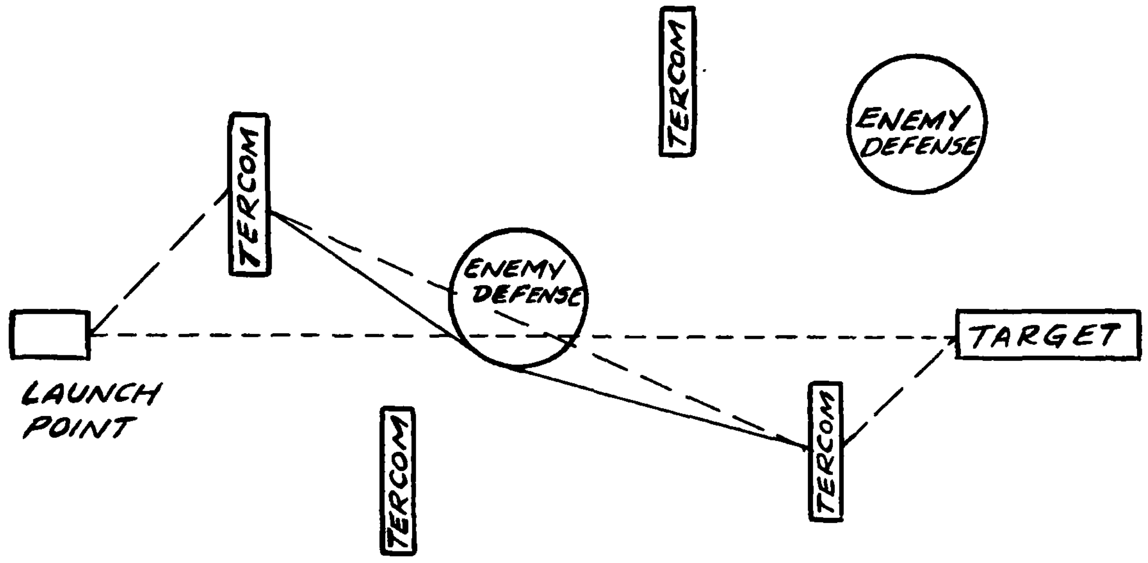


Figure No. 15 - Human Planning Of An ALCM Mission Plan

Once the flight path is proposed, the route is examined and those TERCOM maps closest to the flight path satisfying any constraints on TERCOM map usage are selected for use by this mission. Once the TERCOM maps are selected, a second flight path can be proposed that will overfly the selected TERCOM maps. Thus, if n TERCOM maps are selected, the new flight path will consist of n+1 route segments. Since the optimal flight path between any two points is a direct route each of the n+1 route segments is a direct route between the two points used to define that route segment. In Figure No. 15, the flight path represented by the dashed line is the proposed flight path after considering TERCOM map usage.

Once the new flight path has been proposed, it can be examined to ensure minimal exposure to any enemy defensive sites. If the flight path does not overfly any defensive sites then it is used for the mission plan. However, if a defensive site is overflown then an alternative flight path should be proposed. This new flight path would be a modification of the previous flight path where the defensive site previously overflown is bypassed in the new flight path. In the example shown in Figure No. 15, this modification to the flight path is shown as a solid line. This process continues in an iterative fashion until the latest alternative proposed satisfies all the constraints and minimizes the vehicles exposure to enemy defensive sites. Each proposed alternative modifies the previous alternative to correct a specifiec constraint violation or to bypass a specific enemy defensive site.

This is the general type of approach a mission planner would use when planning ALCM missions. While different individuals may vary the specific sequencing of proposed flight paths and the ordering of constraints to be considered, this basic approach will be used by all mission planners. The conclusion reached from this is that mission planners do not generate and evaluate all possible alternatives; selecting the best after all the alternatives have been considered. Many alternatives are simply rejected upfront as infeasible. And, just as a human mission planner does not evaluate all possible alternatives, a computer system performing mission planning should not evaluate all possible alternatives. Instead, only those alternatives which are feasible should be considered for evaluation. This belief was the driving force behind selecting a hierarchical approach to planning when designing the MPLANNER system.

MPLANNER Levels Of Abstraction

Since MPLANNER is a hierarchical planner, understanding the levels of abstraction used is critical to understanding how MPLANNER functions. As was pointed out in chapter four, a planning problem can be reduced to a graph search problem if the state space and operators used to represent the planning problem are defined in sufficient detail. This concept is central to being able to implement the design of MPLANNER and was used when implementing the prototype MPLANNER software. Therefore, each of the three levels of abstraction used by MPLANNER are clearly defined in terms of the state space and operators available at each abstraction level.

State Spaces. An important concept of a hierarchical planner is that the state space defined for a given level of abstraction is a subset of that state space defined for the next lower level of abstraction. This is because, as was pointed out in chapter four, each state space represents a unique set of facts about the problem domain. The higher the level of abstraction, the few of these facts that are being considered.

At the top abstraction level the factors to be considered deal with rather an attempt to plan the mission should be made. Therefore, the state space consists of only those states necessary to define the flight path of the current alternative. The first state represents the coordinates of the launch point and the last state represents the coordinates of the target. Any other states represent the coordinates where heading changes occurred during the flight.

At the middle abstraction level, the factors involved with the proper usage of the TERCOM maps are considered. Thus, the state space for this abstraction level consists of the states from the top abstraction level, plus one state per TERCOM map defined in the MPLANNER database. Each state associated with a TERCOM map represents the coordinates of the center of that TERCOM map.

At the bottom abstraction level, the additional factor to be considered is enemy defenses. Thus, the state space for the bottom level of abstraction consists of those states defined for the middle abstraction level plus a set of eight states for each of the enemy defensive sites in the MPLANNER database. Those states associated with a defensive site represent the coordinates of a set of points that surround the defensive site just beyond it's maximum range. These points are located on consecutive radials, in 45 degree increments, from the defensive site as is shown in Figure No. 16.

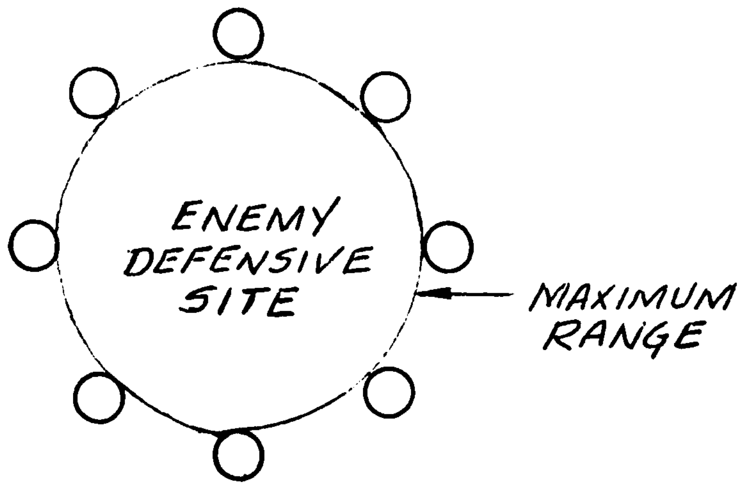


Figure No. 16 - State Spaces Per Enemy Defensive Site

Operators. A standard set of 360 operators are available at each abstraction level. These operators represent the vehicle leaving a point (the coordinates of a defined state in the state space) on a given heading. There is one operator defined for each possible heading that can be expressed in integer degrees. Each operator is subject to the precondition that another state must lie on that heading represented by the operator. Both the state the operator is applied to and the resulting state must be members of the same state space. Due to this precondition, there exists only one usable operator per state space at the top level of abstraction and the number of usable operators available increases with each lower level of abstraction.

In addition to the precondition which the operators are subject to, each operator is also subject to constraints that vary with the level of abstraction. At the top level of abstraction there are two constraints which each operator must satisfy. First, the length of the flight path that results from applying the operator must not exceed the range capability of the vehicle. And, second, any TOA constraint specified by the mission planner must be satisfiable using the flight path which results from applying the operator.

At the middle level of abstraction, the operators are subject to constraints which ensure the proper usage of the available TERCOM maps. In a production version of MPLANNER, these constraints would be established by the mission planners. However for this thesis, constraints were selected

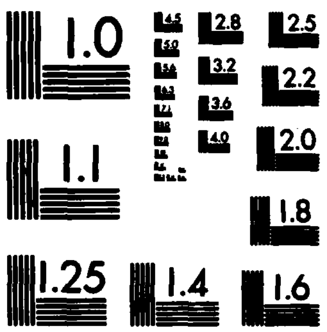
AD-A154 466 AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING AIR LAUNCHED CRUI..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. R J MILLAR 2/2

UNCLASSIFIED DEC 84 AFIT/GCS/ENG/84D-17 F/G 9/2 NL

END

FILMED

DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

that were thought to be reasonable - given the ALCM navigation requirements. In this thesis, all TERCOM maps are classified as either a LANDFALL, ENROUTE, or TERMINAL TERCOM. The constraints on the use of these TERCOM maps requires a mission plan utilize one TERCOM map from each category. The first TERCOM map used must be a LANDFALL TERCOM, the second a ENROUTE TERCOM, and the third a TERMINAL TERCOM map.

At the bottom level of abstraction, the operators are not subject to any constraints in this thesis. However, in a production mission planning system the mission planners may be aware of some constraints that would prevent totally unfeasible flight paths from being considered.

By clearly defining the state space and operators to be used at each abstraction level, the planning problem can be mapped into a graph search problem. The state space for each abstraction level is represented as a seperate graph. The nodes of the graph represent the states in that state space and the links from each state represent the operators that can be applied to that state.

Measure-Of-Merit

Each graph representing the state space of an abstraction level can be searched and one or more paths found from the start node (the launch state) to the goal node (the target state). However, to find the optimal path requires some measure-of-merit be available for assigning a realistic cost to each path. The optimal path is then defined as that path from the start node to the goal node with the lowest cost.

A measure-of-merit should realistically reflect the desirability, or cost, of a particular alternative. In the mission planning domain, three primary factors determine the desirability of a given flight path - attrition, vehicle survival, and navigation error. Therefore, the measure-of-merit used by MPLANNER is computed by summing penalty points that are assessed against the flight path by these factors. Each penalty point represents a percentage point reduction in the probability of the ALCM reaching the target.

Attrition. The attrition factor embodies the concept of a direct flight being optimal. By assessing a penalty for each mile of a flight path, the direct flight path will always have the fewest penalty points. However, the attrition penalty points must be in perspective with the penalty points assessed the flight path by the other factors used in the measure-of-merit. Otherwise, a flight path directly overflying a enemy defensive site could be assessed fewer penalty points than the flight path which bypassed that same defensive site - simply because it was shorter.

MPLANNER assesses a flight path 0.01 penalty points per mile for attrition. This number was selected because it is reasonable and easy to calculate. In a production mission planning system, it is likely that the mission planners would select a different value for attrition.

Vehicle Survival. The second factor considered in the measure-of-merit is the survival of the vehicle. The more likely that a flight path would result in the vehicle being destroyed enroute, the less desirable that flight path becomes. Therefore, in MPLANNER, when a flight path traverses the threat radius of an enemy defensive site, it is assessed penalty points based upon the type of defensive site.

<u>Navigation Error</u>. The other factor included in the measure-of-merit is navigation error. A flight path which is likely to generate navigation errors, resulting in the vehicle becoming lost, is undesirable. In MPLANNER, this factor is considered by evaluating the flight paths and assessing penalty points based upon the likelyhood of the vehicle developing navigation problems enroute to the target.

These three factors - attrition, vehicle survival, and navigation error - are the primary items to be considered when evaluating one flight path against another. By combining these factors into a single measure-of-merit the MPLANNER system has simple, but realistic, method of comparing multiple flight paths and identifying the best one.

<u>Estimating Flight Path Costs</u>

In addition to realistically appraising the desirability of a flight path, the measure-of-merit used by MPLANNER also forms the basis of the heuristic used to estimate flight path costs at the top abstraction level. When estimating the cost of a flight path, only the attrition factor of the measure-of-merit is used. This makes an excellent heuristic as the estimate is never greater than the true cost of the flight. This guarantees, as was discussed in chapter four, that the optimum flight path will not be inadvertently discarded.

MPLANNER Approach To Planning

The basic approach used by MPLANNER for mission planning is very similiar to the approach used by people. Beginning with a direct flight path, MPLANNER goes through a process of evaluating flight paths. Any flight path violating a constraint is replaced with alternative flight paths that do not violate that constraint. Once a set of flight paths have been identified that satisfy all the constraints, the best alternative can be selected for proposal to the mission planner.

This process is begun by MPLANNER considering the direct flight path from launch point to target at the top abstraction level. If one of the constraints at this abstraction level is violated then the requested mission plan can not be prepared. However, if the top abstraction level constraints are satisfied then an estimated cost is assigned to the direct flight path and it is placed on a list of alternatives that MPLANNER maintains. This list, and the costs associated with each alternative on it, are key elements in the planning process.

Once the alternative list has been initialized, an iterative process is begun of evaluating alternative flight paths. Each iteration begins with selecting the lowest cost flight path from the alternative list. Since the direct flight path is the only alternative initially on the list it is obviously the first alternative considered. When selecting an alternative, one of three possible conditions will occur - the alternative list will be empty, the alternative selected was previously considered at the bottom abstraction level, or the alternative was previously considered at only the top abstraction level.

If the alternative list is empty then no flight path exists, from the launch point to target, which satisfies all the constraints. In this case, MPLANNER terminates it's planning attempts and advises the mission planner that a suitable flight path could not be identified. The planning process is also terminated when the selected alternative has been previously considered at the bottom level of abstraction. In this case, the selected alternative is the least cost flight path which satisfies the constraints. This alternative is then suggested to the mission planner as the planning process is terminated. To understand why these conditions result in termination of the planning process, it is necessary to understand the processing which takes place when the third condition occurs.

When the flight path selected has been previously considered at only the top level of abstraction then it is considered at the middle abstraction level and one of two possibilities will occur. If the selected flight path satisfies all the constraints applied at the middle abstraction level then processing continues to the bottom level of abstraction. At this level, the flight path is evaluated and assigned a true cost based upon the measure-of-merit described earlier in this chapter. The flight path is then added back to the list of alternatives labeled with it's true cost. If any penalty points were assessed the flight path by either the vehicle safety or navigation error factors of the measure-of-merit, then MPLANNER attempts to suggest alternative flight paths. These suggested alternatives are designed to reduce the number of penalty points assessed the new flight paths by identifying that route segment of the selected flight path which received the most penalty points and replacing it with two route segments that attempt to avoid whatever caused the penalty points to be assessed. After the alternative flight paths have been suggested, the next flight path is selected from the alternative list.

The other possibility when considering the selected flight path at the middle abstraction level is that the flight path does not satisfy all the constraints. In that case, the flight path is discarded from further consideration and one or more alternative flight paths are suggested, if possible, by MPLANNER. Each alternative suggested by MPLANNER is composed by adding one additional node and two links to the graph of the discarded flight path. This effectively replaces that route segment guilty of violating the constraint with two route segments designed to ensure the constraint is satisfied by the new flight path. Once all possible alternatives have been suggested, another flight path is selected from the alternative list.

When MPLANNER suggests an alternative flight path it is immediately considered at the top level of abstraction. If the constraints applied at that abstraction level are not satisfied, the suggestion is simply ignored by the system. Otherwise, it is assessed an estimated cost and added to the list of alternatives to be considered. This guarantees that every flight path on the alternative list has been successfully considered at the top level of abstraction. Another benefit of this approach is that an estimated cost does not need to be calculated for every alternative suggested by MPLANNER - only those which satisfy the constraints applied at the top abstraction level.

This approach to planning utilizes a combination of techniques to reduce the number of alternatives considered and the amount of detail contemplated while considering an alternative. The propagation of constraints at the top two levels of abstraction allows many alternatives to be quickly eliminated from further consideration. The attrition factor of the measure-of-merit, as a cost estimator, permits a heuristic search to quickly identify a flight path which satisfies all the constraints. Once the true cost of that flight path is known, the cost can be used as an upper bound for other alternatives. The moment the estimated cost, or true cost, of an alternative exceeds this upper bound it can be eliminated from further consideration. If a flight path is found with a lower true cost, that cost becomes a better upper bound. These combined techniques allow MPLANNER to achieve its goal of considering a minimal set of alternatives while still finding the best possible flight path available.

Route-Planning-KS

The mission planning component of MPLANNER implements the planning approach just described. As was discussed in chapter five, a modified meta-planning control structure is used to direct the planning process. As the design space, the Route-Planning-KS is the main driver for the mission planning component and controls the planning process. It suggests alternative flight paths, determines when the planning process is complete, and does the actual planning. The other two knowledge sources are used for computing components of the measure-of-merit assigned each alternative at the bottom level of abstraction in the planning process.

When the Route-Planning-KS finds a "PLAN-SORTIE" message on the message blackboard, it awakens from the sleep mode and begins the planning process by initializing the alternative list to a direct flight path from the launch point to the target specified for that sortie. Then the iterative process described earlier begins. If a flight path can not be generated which satisfies all the constraints, then the Route-Planning-KS posts a "PLAN-ABORT" message to the message blackboard and the planning process for that sortie is terminated. Otherwise, when the optimum flight path is identified, a "PLAN-COMPLETE" message is posted to the message blackboard when the planning process terminates. In either case, once the planning process is terminated for the sortie, the Route-Planning-KS goes back to sleep until another "PLAN-SORTIE" message is posted to the message blackboard.

Whenever the Route-Planning-KS needs to assign a flight path a measure-of-merit, it posts a "EVALUATE" message to the message blackboard. Once this message is posted, the flight path is simultaneously evaluated by the Route-Evaluation-KS and the Vehicle-Performance-KS knowledge sources. As each KS completes it's evaluation of the flight path, they post a "EVALUATION-COMPLETE" message to the message blackboard - along with the results of their evaluation. When the Route-Planning-KS receives both copies of the "EVALUATION-COMPLETE" message a measure-of-merit can then be assigned to the flight path and the planning process continues.

Evaluating a flight path is computationally the most expensive function performed by MPLANNER. The planning approach used reduces the number of evaluations required to plan a mission. When a evaluation is necessary it is done in parallel - capitalizing on the parallel processing potential of a dual blackboard system. These two factors result in greatly reduced computational costs for planning a mission.

<u>Route-Planning-KS Architecture</u>. In chapter six, it was pointed out that many AI concepts were used in the design of MPLANNER. However, the Route-Planning-KS is the only KS which is clearly an AI expert system with the classic seperation of knowledge base and inference engine. The inference engine for this KS is an interpreter for a rule based production system as was described in chapter four. The knowledge base is a collection of IF-THEN production rules describing the rules used during the planning process.

A production system architecture was selected for this KS because the knowledge required for planning a mission is extensive and continually evolving. Since the mission planners are the only people with this knowledge, they must be able to transfer that knowledge to the computer system and be able to readily modify that knowledge. The simple IF-THEN format of production rules facilitates this transfer of knowledge and can be easily maintained by the mission planners with minimal training. These rules will identify the set of facts used to define potentially undesirable aspects of a flight path and which facts need to be changed to make that aspect of the flight path more desirable. These rules will then be used by MPLANNER during the planning process when considering flight paths and suggesting alternatives.

With the mission planners specifying the rules, MPLANNER can be thought of as more of a planning tool, akin to a spreadsheet program, than a blackbox planning system. This should improve the acceptance given the system by the mission planners as they will actually control MPLANNER - versus the system controlling how they plan the missions. This is very important as mission planners are extremely cautious about anything dealing with vehicles that carry nuclear warheads.

In addition, the mission planners building the rule database will improve the quality of the mission plans generated. This is because the quality of the rules used by the planning process dictates the quality of the mission plans generated. By having the mission planners develop these rules, the rules are expected to gradually evolve into a robust set which includes the knowledge of many of the best mission planners. When this occurs, the quality of a mission plan proposed by MPLANNER should equal (or maybe exceed) the quality of a mission plan developed by the mission planners.

<u>Route-Evaluation-KS</u>

When the Route-Planning-KS posts an "EVALUATE" message to the message blackboard, the Route-Evaluation-KS awakens from a sleep mode and evaluates the indicated flight path with respect to vehicle survival. Once the flight path has been evaluated, an "EVALUATION-COMPLETE" message is posted to the message blackboard. This message includes the number of penalty points to be assessed the flight path and why the penalty points were assessed. Once the message is posted to the message blackboard, the Route-Evaluation-KS reenters a sleep mode and waits for another message.

<u>Vehicle-Performance-KS</u>

When a "EVALUATE" message is posted to the message blackboard the Vehicle-Performance-KS awakens from a sleep mode and evaluates the indicated flight path with respect to navigation error. Once the flight path is evaluated, the KS posts a "EVALUATION-COMPLETE" message to the message blackboard and goes back to sleep. This message includes the number of penalty points assessed the flight path due to navigation error and the reason the penalty points were assessed against the flight path.

Summary

The mission planning component is the very heart and soul of the MPLANNER system as it performs the actual mission planning. Three knowledge sources make up the planning component of the system, but the Route-Planning-KS is where the mission planning is done. The other two knowledge sources are used only for evaluating promising flight paths.

A combination of techniques are used to reduce the number of alternative flight paths that must be considered before the optimal flight path is identified. The key concept behind these techniques is simply to consider only those factors that will impact the success or failure of the mission plan. When a promising flight path is identified, the evaluation of that flight path is done in parallel. These two factors reduce the computational costs of planning a mission plan to the point where each sortie can be planned seperately - reflecting the dynamic nature of the problem domain.

VII. Summary And Conclusions

This thesis has proposed a specific design for a computer system capable of suggesting ALCM mission plans to a mission planner. This proposed design is based upon recent research in the field of AI and has three major features of significance. These are the dual blackboard architecture, the rule based production system for controlling the planning process, and the unique approach to planning utilized to reduce the number of alternatives that require evaluation.

Dual Blackboard Architecture

A dual blackboard architecture was selected for this design for two primary reasons. First, it forces the utilization of several key software engineering concepts. These concepts enhance the probability of the system performing as expected after it has been developed. These concepts also enhance the maintainability of the system after development.

The second reason for selecting a dual blackboard architecture was because this architecture can apply the power of parallel processing to the problem domain if implemented on a distributed computer system. However, the development of the system does not need to be done on a distributed computer system as a dual blackboard can also be implemented on a single computer - if it supports multiple processes. This reduces the risk associated with developing MPLANNER because parallelism inherent in the problem domain can be utilized in the design without the expense of procuring a distributed system to develop the system on. Once MPLANNER has been developed and demonstrated to perform properly, then it can be implemented on a distributed system when cost effective.

Rule Based Production System

The use of a rule based production system for the Route-Planning-KS allows the mission planners to control how the MPLANNER system performs the planning functions. Therefore, this proposed design is more along the lines of an ALCM planning tool for use by the mission planners; instead of a computer system designed to replace the mission planners. The implication of this is that the mission planners are responsible for the quality of the mission plans suggested by MPLANNER. The better the production rules provided; the higher the quality of the mission plans. The MPLANNER design guarantees only the computational costs associated with planning a mission; not the quality of the mission plan.

Therefore, it should be expected that the quality of the mission plans suggested by MPLANNER will improve with time. Initially, the mission plans are likely to be unacceptable as the mission planners learn how to write production rules and develop a basic set of rules about planning ALCM missions. It may require several revisions of the production rules before the suggested mission plans become acceptable. However, over the long term it is expected the set of production rules used will become quite robust and the quality of the suggested mission plans will become very high.

MPLANNER Approach To Planning

MPLANNER uses a propose, critique, and fix-up approach to planning. The most promising flight path is proposed for consideration - beginning with the direct flight path from launch point to target. The proposed flight path is then critiqued by assessing the flight path penalty points for undesirable characteristics. Based upon that critique, more flight paths are proposed as the system attempts to fix-up the flight path by modifying the route segment which was assessed the most penalty points. The flight path that is suggested for the mission plan is that flight path which best approximates a direct flight path while satisfying the appropriate constraints and has been assigned minimal penalty points.

This approach to planning is implemented with a hierarchical planner, combined with several other techniques, to minimize the number of alternative flight paths that require evaluation before the optimum flight path can be identified. Those alternatives not satisfying all constraints associated with ALCM missions are rejected upfront with very minimal computational costs. Of those alternative flight paths that do satisfy all the constraints, only those most promising are evaluated. Hueuristics are used to determine if a flight path is promising, but the heuristics are constrainted well enough to ensure the optimum flight path is not overlooked. The result of this approach to planning is that the computational cost of planning a mission is extemely inexpensive.

Areas For Further Study

While the design proposed for MPLANNER is an excellent beginning, additional study is required before a production version could be developed. The biggest discrepancy with the proposed design is the use of only three abstraction levels. It is obvious that several additional levels of abstraction would be necessary for a production system. For example, the next lower abstraction level (after the three defined in this thesis) might use state spaces representing the coordinates of landmasses having radical changes in altitude. Thus, clobber could be considered at this abstraction level. The issue of additional abstraction levels requires considerable further study. Otherwise, an arbitrary selection of abstraction levels could seriously undermine the effectiveness of this proposed design.

In addition to considering the number of abstraction levels necessary for a production system, the concept of planning to variable abstraction levels needs to be studied. It is intuitively obvious that not every mission plan needs to be planned to the bottom level of abstraction. But, how can it be avoided? Possibly, indicators of terrain roughness and density of defenses could be calculated for each level of abstraction. Then the planning process could proceed down the abstraction levels until an abstraction level is reached whose indicators are below a given value. Planning to variable abstraction levels appears to be the most promising enhancement which could be made to this proposed design.

Conclusions

The design proposed in this thesis for a ALCM mission planning system is an excellent framework around which a production system can be developed. The major enhancement required is to add more levels of abstraction to the planning process. The precise number of abstraction levels needed is subject to debate, but the top three levels have already been defined and the bottom level should be consistent with the level of detail currently used by the mission planners when they evaluate mission plans manually generated.

Therefore, when the decision is made to develop an ALCM mission planning system the author believes this thesis should be presented to the contractors involved as a suggested design. In the process of verifying the design proposed in this thesis, the contractors will probably find several ways of improving the design. This improved design should then be used to develop the mission planning system.

Bibliography

1. Barr, Avron and Edward A. Feigenbaum. The Handbook of Artificial Intelligence (Volume 1). Stanford, Ca.: Heuristech Press, 1981.

2. Barr, Avron and Edward A. Feigenbaum. The Handbook of Artificial Intelligence (Volume 2). Stanford, Ca.: Heuristech Press, 1981.

3. Cohen, Paul R. and Edward A. Feigenbaum. The Handbook of Artificial Intelligence (Volume 3). Stanford, Ca.: Heuristech Press, 1981.

4. Correll, John. "Deterrence Today," Air Force Magazine, 67:40-45 (February 1984).

5. Davis, Randall. "Expert Systems: Where Are We? And Where Do We Go From Here?," The AI Magazine, 3:3-22 (Spring 1982).

6. Gevarter, William B. "Expert Systems: limited but powerful," IEEE Spectrum, 20:39-45 (August 1983).

7. Hayes-Roth, Frederick and others. "An Overview Of Expert Systems," Building Expert Systems, edited by Frederick Hayes-Roth and others. London: Addison-Wesley Publishing Company, 1983.

8. Heller, Warren G. and Richard LeSchack. Military Geodesy And Geospace Science (Unit Two). Contract F19628-77-C-0152. The Analytic Sciences Corporation, Reading Mass., February 1981 (AD-A105 071).

9. Jardine, T.J. and S.T. Shebs. "Knowledge Representation In Automated Mission Planning," Proceedings of The International Joint Conference on Artificial Intelligence, 1:9-14 (1983).

10. Marsh, J., Chief, Cruise Missile Path Optimization Program. Program Review. Systems Control Technology Corp., Palo Alto, Ca., 14 September 1982.

11. Robinson, Clarence A. "USAF Planning Stealth Cruise Missile," Aviation Week & Space Technology, 117:18-21 (November 8, 1982).

12. Rogers, Patricia R. "ALCM In It's Second Operational Year," Air Force Magazine, 67:46-48 (February 1984).

13. Stefik, M. J. Planning With Constraints. PhD dissertation. Stanford University, Stanford Ca., 1980 (Rep. No. 80-784).

14. Taylor, John W., et. all. Jane's ALL THE WORLD'S AIRCRACT, 1977-1978. New York: Macmillan Publishing Co., 1978.

15. Toomay, John C. "Technical Characteristics," CRUISE MISSILES: Technology, Strategy, Politics, edited by Richard K. Betts. Washington, D.C.: The Brookings Institute, 1981.

16. Tsipis, Kosta. "Cruise Missiles," Scientific America, 236:20-29 (February 1977).

17. Wellborn, Stanley N. "Machines That Think," U.S. News & World Report, 99:59-62 (December 5, 1983).

18. Wilkins, David E. "Domain-Independent Planning: Representation and Plan Generation," Artificial Intelligence, 22:269-301 (April 1984).

19. Winston, Patrick H. Artificial Intelligence. Menlo Park: Addison-Wesley Publishing Company, 1979.

Captain Robert J. Millar was born to Jim and Romona Millar on 28 September 1951 in Pierre, South Dakota. He graduated from the Pierre high school in 1969 and enlisted in the USMC in June of that year. After several assignments as a COBOL programmer, he seperated from the USMC in 1977 to attend school at the University of Nebraska - Lincoln. Two years later, in May 1979, he received the degree of Bachelor of Science in Computer Science. Upon graduation, he received a commission into the USAF through the ROTC program. Shortly thereafter, August 1979, he was assigned to Headquarters, Strategic Air Command at Offutt AFB, Nebraska. There he perfomed duty as a systems analyst with the weapons effects section of the Directorate of War Plans Programming. His primary duties involved computer simulation of strategic communications in a nuclear environment. In May 1983, he entered the School of Engineering, Air Force Institute of Technology at Wright-Patterson AFB, Ohio.

Permanent Address: 210 S. Filmore
Pierre, South Dakota
57501

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE:

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GCS/ENG/84D-17

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: Air Force Institute of Technology

6b. OFFICE SYMBOL *(If applicable)*: AFIT/EN

7a. NAME OF MONITORING ORGANIZATION:

6c. ADDRESS *(City, State and ZIP Code)*: Wright Patterson AFB, Ohio 45433

7b. ADDRESS *(City, State and ZIP Code)*:

8a. NAME OF FUNDING/SPONSORING ORGANIZATION:

8b. OFFICE SYMBOL *(If applicable)*:

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c. ADDRESS *(City, State and ZIP Code)*:

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*: See Block 19

12. PERSONAL AUTHOR(S): Millar, Robert Joseph

13a. TYPE OF REPORT: MS Thesis

13b. TIME COVERED: FROM ________ TO ________

14. DATE OF REPORT *(Yr., Mo., Day)*: 1984 December

15. PAGE COUNT: 100

16. SUPPLEMENTARY NOTATION:

Approved for public release: IAW AFR 190-17.
[illegible]
LYNN E. WOLAVER 11 April 85
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES

FIELD	GROUP	SUB. GR.
06	04	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*: Artificial Intelligence; Planning; Computer Programs

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

AN ARTIFICIAL INTELLIGENCE BASED FRAMEWORK FOR PLANNING AIR LAUNCHED CRUISE MISSILE MISSIONS - This report describes a design for a computer system capable of generating mission plans for an Air Launched Cruise Missile (ALCM). This design is based upon recent research in the field of Artificial Intelligence and has three major features. A hybrid blackboard architecture, called a dual blackboard, is developed in this report as the basic architecture of this planning system. This hybrid architecture emphasizes the parallel processing potential inherent in a blackboard system. In addition, it forces some structure onto the blackboard concept - preventing a blackboard from being implemented as simply a global variable or common block.

The knowledge source performing the actual mission planning is designed as a rule based production system. This allows the mission planners to build the rules used for the planning process. These rules define for the system what makes a mission plan undesirable and what can be done to make that mission plan more acceptable. As the mission planners

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION: Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL: Capt Steven Cross

22b. TELEPHONE NUMBER *(Include Area Code)*: 513-255-5533

22c. OFFICE SYMBOL: AFIT/EN

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

specify the rules used, this system should be viewed as more of a ALCM planning tool, akin to a spreadsheet program, instead of a blackbox system used for planning ALCM missions.

A modified meta-planning approach to planning is used in conjunction with several other techniques to reduce the number of alternative mission plans that must be evaluated before the best plan can be identified. Domain specific constraints are applied against mission plans proposed by the system to ensure only those mission plans satisfying all constraints are even considered for evaluation. Heuristics are used to rank those mission plan alternatives satisfying all the constraints; ensuring only the most promising get evaluated. The heuristic used in this design favors those mission plans that best approximate a nominal mission plan - one with a direct flight path.